

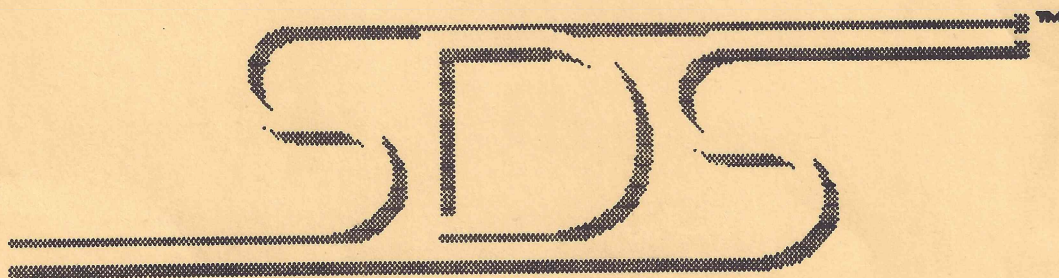
Southwestern Data Systems

Presents

the Wonderful All New



"and documentation thereof"

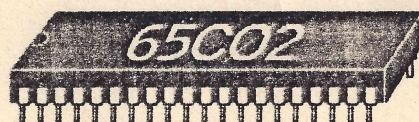


© 1983 Southwestern Data Systems

Southwestern Data Systems

Presents

the Wonderful All New

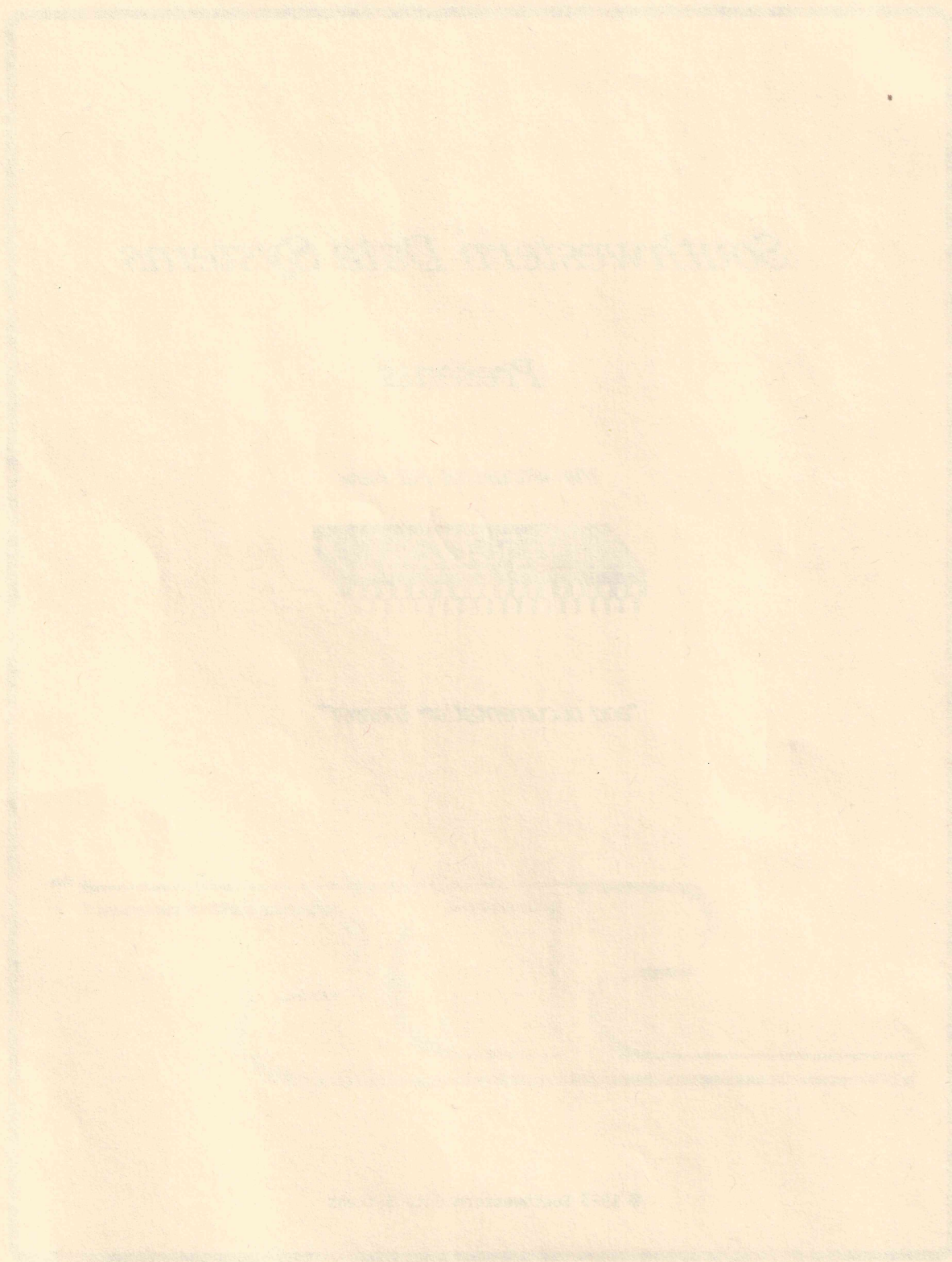


"and documentation thereof"

SDS™

© 1983 Southwestern Data Systems

RAWP cover



DOCUMENTATION OVERVIEW

This documentation has been written for those who are already acquainted with 6502 assembly language and wish to merely learn the operations and addressing modes that have been added to the 6502. To realize the potential of this chip, it is recommended that you write or acquire software that takes advantage of the powerful additional new features of the 65C02.

SECTION ONE

This section is a brief tutorial on the new instructions.

SECTION TWO

This section is a summary of the new instructions, containing the instruction name, syntax, opcode, number of bytes and number of cycles.

SECTION THREE

This section is an updated and enhanced table of the 65C02 instruction set, modeled after the table that appears in the Apple II Reference Manual. This should be a handy quick reference guide suitable for access during programming.

Note regarding
65C02 GTE
vs Rockwell

This document is a copy of a letterhead memorandum (LHM) dated 10/1/68, from the Director of the Central Intelligence Agency (CIA) to the Chief of the Bureau of the National Security Agency (NSA). The LHM is classified "Secret" and contains information that is not to be released to the public.

SECRET

This document is a copy of a letterhead memorandum (LHM) dated 10/1/68, from the Director of the Central Intelligence Agency (CIA) to the Chief of the Bureau of the National Security Agency (NSA).

SECRET

This document is a copy of a letterhead memorandum (LHM) dated 10/1/68, from the Director of the Central Intelligence Agency (CIA) to the Chief of the Bureau of the National Security Agency (NSA). The LHM is classified "Secret" and contains information that is not to be released to the public.

SECRET

This document is a copy of a letterhead memorandum (LHM) dated 10/1/68, from the Director of the Central Intelligence Agency (CIA) to the Chief of the Bureau of the National Security Agency (NSA). The LHM is classified "Secret" and contains information that is not to be released to the public.

DISCLAIMER

SOUTHWESTERN DATA SYSTEMS SHALL HAVE NO LIABILITY OR RESPONSIBILITY TO PURCHASER OR ANY OTHER PERSON OR ENTITY WITH RESPECT TO ANY LIABILITY, LOSS OR DAMAGE CAUSED OR ALLEGED TO BE CAUSED DIRECTLY OR INDIRECTLY BY THIS HARDWARE, INCLUDING, BUT NOT LIMITED TO ANY INTERRUPTION OF SERVICE, LOSS OF BUSINESS OR ANTICIPATORY PROFITS OR CONSEQUENTIAL DAMAGES RESULTING FROM THE USE OR OPERATION OF THIS HARDWARE.

A Word About Software Compatibility

The new 65C02 will not operate correctly on the older Apple II's and II+'s due to slightly different timing requirements.

While this 65C02 is downward software compatible with the 6502, a few programs may use unsupported opcodes of the 6502. These programs will not run on the 65C02.

Plugging in the 65C02 will not alter the speed of most 6502 programs. The only difference we can tell at this point is that decimal addition and subtraction take an extra clock cycle. Any initial benefits of installing the 65C02 will be the result of the users own machine language programming. Southwestern Data Systems MERLIN assemblers will accommodate the additional instructions of the 65C02.

or Merlin PRO

OR commercial programs that take advantage Blah Blah

INSTALLATION INSTRUCTIONS Apple IIe

1. Turn off the power.
2. Remove the cover.
3. Locate the 6502 at location B4.
It is a good idea to remove all peripheral cards from their slots at this time. CAUTION: avoid all sources of static electricity when removing and installing cards and chips. It is a good idea to continually discharge any possible static build up by touching the power supply case inside the Apple.
4. Remove the 6502 (it has 40 pins) from location B4 by gently prying each end up with a small screwdriver. Be sure to not pry against the motherboard. A chip puller works better if you have one. Also note the orientation of the notch on the end of the 6502 chip. It should be toward the keyboard side of the computer.
5. Place the 6502 aside and remove the 65C02 from its protective foam. Insert the 65C02 in the empty socket at location H3-H5 orienting it the same way as the 6502 previously removed (notch toward the keyboard).
6. Check for bent pins by carefully examining the seated chip. If bent pins are found carefully remove the 65C02. Straighten the bent pins and reinsert the 65C02 again. The pins can probably be straightened once or twice.
7. Place the removed 6502 in the protective foam and save in case of problems with the 65C02.
8. Now is a good time to verify operation of the 65C02 by powering up the computer before reinstalling the peripheral cards in their slots and verifying normal operation.
9. Replace the removed peripheral cards and the lid. Power up to verify normal operation and then enjoy the new instructions of the 65C02.

Section One

65C02 Tutorial

Make a table of the values of the counter when the 65C02 starts. I'll be back.

Use
fund for I

Note
GTE is the model
that comes with Apple's Enhanced Ke Kit

SECTION ONE

A DETAILED INTRODUCTION TO THE 65C02

Delete This documentation deals with a new version of our beloved 6502 microprocessor known as the "65C02". Although the chip has just been released within the last few months, and as such has yet to find its way into the mainstream of computers, it seems likely that we'll be hearing more about this item in the upcoming year. The chip can be plugged right into existing Apple computers and is compatible with existing software.

Before jumping right into the new functions though, let's first get a little background information out of the way.

Re-write The 6502 was apparently originally designed by Commodore computers, and as of the present, 70% of their use is by Apple, Atari and Commodore. The current manufacturers of the 6502 are Rockwell International, MOS Technology and Synertek. As it happens with these things though, some of the key persons involved with the 6502 went to work at a new company, Western Design Center. This company then is the original source of the new 65C02 chip. But the story doesn't end there. Western Design Center has sold the design to at least three independent manufacturers, Rockwell International, GTE, and NCR. Each of these companies took the initial 65C02 design, corrected initial design errors, and added their own enhancements.

[The picture at this point is that each of these three companies will be marketing their own version of the 65C02.] The chips are more or less the same, but the Rockwell chip has the largest instruction set.

"Largest instruction set?", you ask? Yes! The new 65C02 has had the old instruction set increased to add a rather large variety of new instructions. Because the Rockwell chip appears to be a superset of all the other chips, the bulk of this text will assume that is the chip that is being used. At the end of this documentation I'll describe the individual differences that I've been able to discover.

The Rockwell chip has a total of 12 new instructions, and two new addressing modes. In addition a number of addressing modes not normally available to an instruction (such as the immediate mode for the BIT instruction) are now available. There are a total of 59 actual new opcodes. The meaning of all these numbers will become clear shortly.

NEW ADDRESSING MODES

Since this is one of the smaller numbers, let's start here. You'll recall from many earlier discussions that each 6502 instruction has a possible of six addressing modes. We will assume, for purposes of simplicity, that a few addressing modes are merely variations of one another, and I am also not including the value associated with relative branch instructions (BEQ, BNE, BCC, BCS, etc.) as an addressing mode here. To refresh your memory, a short example is provided here for the LDA (Load Accumulator) instruction.

Addressing Mode	Common Syntax	Hex Coding
1. Absolute	LDA \$1234	AD 34 12
Zero Page	LDA \$12	A5 12
2. Immediate	LDA #\$12	A9 12
3. Absolute,X	LDA \$1234,X	BD 34 12
Zero Page,X	LDA \$12,X	B5 12
4. Absolute,Y	LDA \$1234,Y	B9 34 12
5. (Indirect,X)	LDA (\$12,X)	A1 12
6. (Indirect),Y	LDA (\$12),Y	B1 12

INDIRECT ADDRESSING

The first of the two new addressing modes is quite easy to explain because it is essentially a variation of an existing instruction. The new mode is "Indirect" addressing. This may sound very familiar because this is similar to the instructions used to access memory locations via a zero page pointer. Usually, though, the Y register is set to zero or some other value, which is then added to the address indicated by the zero page pointer to determine the address we wish to access.

This is fine for accessing a large table of data, but many times we are interested in only one byte of memory, and must then go through the obligatory 'LDY #\$00' (or 'LDX #\$00') to properly condition the Y (or X) register. (See table entries #5 and #6 above).

The new instruction allows us to ignore the contents of the Y register and access the memory location directly. This saves two bytes of code for each reference, since the Y register does not have to be directly loaded. If one wanted to scan a block of memory, such as for a table, this instruction can still be used if you are willing to INC or DEC the zero page pointer accordingly.

Addressing Mode	Common Syntax	Hex Coding
7. Indirect	LDA (\$12)	B2 12

This new addressing mode is available for the following instructions:

Instruction & Common Syntax	Hex Coding
ADC (\$12)	72 12
AND (\$12)	32 12
CMP (\$12)	D2 12
EOR (\$12)	52 12
LDA (\$12)	B2 12
ORA (\$12)	12 12
SBC (\$12)	F2 12
STA (\$12)	92 12

INDEXED ABSOLUTE INDIRECT

The second new addressing mode has a name that was obviously not designed with easy recall in mind. Fortunately, this too is a variation on an existing instruction and as such should be easy to remember. There is currently on the 6502 an indexed indirect addressing, which can be more simply referred to as 'pre-indexed' addressing. An example is in item #5 in the first table of instructions. Preindexing meant that the contents of the X register were added to address of the zero page reference BEFORE using the sum of those numbers to determine which zero page pair to use. For example, the instruction:

LDA (\$22,X)

where the X register held the value '4' would actually access bytes \$26,27 to get the final destination address.

This was opposed to indexed indirect, which we can more simply refer to as 'post-indexing'. In post-indexing the value of the Y register is added AFTER the base address is determined. For example, in the instruction:

LDA (\$22),Y

where the Y register holds the value '4', and \$22,23 point to location \$1000, the memory location accessed would be \$1004.

You'll recall also that pre- and post-indexing were limited in their use of the X and Y registers. Pre-indexing could only use the X register and post-indexing only the Y. Before you get too excited in anticipating the possibilities of the new instruction, let me calm you down to say that this much has not changed.

What has changed is that an absolute pre-indexing addressing mode has been added to compliment the indirect JMP. This makes an easy efficient way to create and access JMP tables.

<u>Addressing Mode</u>	<u>Common Syntax</u>	<u>Hex Coding</u>
8. Indexed Absolute Indirect	JMP (\$1234,X)	7C 34 12

For example, suppose you had a command interpreter that accepted a command value between 0 and 2. Previously, such an interpreter could be implemented the following ways:

ACTUAL PROGRAM LISTING:

```

1 *****
2 * OLD COMMAND PROCESSOR #1 *
3 *****
4 *
5     OBJ $1000
6 JMPAD EQU $0A
7 * Let's assume that assume that GETCMD is a
   subroutine at $4000 that gives us a
   number between 0 and 2
1000: 20 00 40 8 ENTRY JSR GETCMD ; GET VAL FROM 0 - 2
1003: 0A 9 ASL
1004: AA 10 TAX
1005: BD 13 10 11 LDA CMDAD,X
1008: 85 0A 12 STA JMPAD
100A: E8 13 INX
100B: BD 13 10 14 LDA CMDAD,X
100E: 85 0B 15 STA JMPAD+1
1010: 6C 0A 00 16 JMP (JMPAD)
17 *
1013: 00 08 18 CMDAD DA CMD1
1015: 00 09 19 DA CMD2
1017: 00 0A 20 DA CMD3
1 *****
2 * OLD COMMAND PROCESSOR #2 *
3 *****
4 *
5     ORG $1000
1000: 20 00 40 6 JSR GETCMD ; GET VAL FROM 0-2
1003: AA 7 TAX
1004: BD 0D 10 8 LDA CMDH,X
1007: 48 9 PHA
1008: BD 10 10 10 LDA CMDL,X
100B: 48 11 PHA
100C: 60 12 RTS
13 *
100D: 07 14 CMDH DFB >CMD1-1 ;COMMAND HIGH
100E: 08 15 DFB >CMD2-1
100F: 09 16 DFB >CMD3-1
17 *
1010: FF 18 CMDL DFB <CMD1-1 ;COMMAND LOW
1011: FF 19 DFB <CMD2-1
1012: FF 20 DFB <CMD3-1
1 *****
2 * NEW COMMAND PROCESSOR *
3 *****
4 *
5 ENTRY JSR GETCMD ; GET VAL FROM 0-2
1000: 20 00 40 5
1003: 0A 6 ASL ; MULTIPLY BY 2
1004: AA 7 TAX
1005: 7C 08 10 8 JMP (CMDTBL,X)
9 *
1008: 00 08 10 CMDTBL DA CMD1
100A: 00 09 11 DA CMD2
100C: 00 0A 12 DA CMD3

```

Notice how much shorter and straight forward the new instruction makes setting up JMP tables.

NEW "STANDARD" ADDRESSING MODES

There are a few instructions that have addressing modes that are just "new to them". For example, one of the most exciting ones are INC and DEC.

Previously, any use of INC and DEC was limited to memory locations. In addition (so to speak), use of the X and Y registers was the only way to maintain a simple loop counter without using a dedicated memory location. The surprise here is that INC and DEC will now work on the accumulator! This is nice because you can now maintain a counter in the accumulator, or even do fudging of flag values, etc. as they are being handled in the accumulator.

Instruction & Common Syntax

Hex Coding

DEC
INC

3A
1A

Note: Some future assemblers may require the somewhat unusual (if not inconvenient) use of DEC A or INC A here as they seem to prefer for the previous LSR, ASL, etc. instructions.

The BIT instruction also allows some additional addressing modes which may prove useful. The BIT instruction previously only supported absolute addressing. That is to say that a directly referenced memory location was used as the value against which the accumulator was operated on.

Addressing Mode

Common Syntax

Hex Coding

Absolute
Zero Page

BIT \$1234
BIT \$12

AD 34 12
A5 12

This is useful for testing a memory location for a given bit pattern, but not directly suitable for testing the bit pattern of the accumulator. For many operations, this means you had to rather artificially load some memory location with the value you wanted to compare to the accumulator.

The new 65C02 supports three new addressing modes for the BIT instruction:

1. Immediate	BIT #\$12	89 12
2. Absolute,X	BIT \$1234,X	3C 34 12
3. Zero Page,X	BIT \$12,X	34 12

(It should be noted that BIT immediate affects the N flag as an AND instruction would. Bit 7 and bit 6 of the immediate data does not get transferred to the N and V flags, however.)

AT LAST, THE REAL "SCOOP"!

Of course, the real question lurking in everyone's mind is "But what are the new instructions?!" The great thing about the 65C02 is that not only are many of the old instructions enhanced, there are a number of absolutely terrific new instructions. To be exact, the number is 12.

The new instructions are:

BBR	Branch on Bit Reset (clear)
BBS	Branch on Bit Set
BRA	Branch Always
PHX	Push X onto Stack
PHY	Push Y onto Stack
PLX	Pull X from Stack
PLY	Pull Y from Stack
RMB	Reset (clear) Memory Bit
SMB	Set Memory Bit
STZ	Store Zero
TRB	Test and Reset (clear) Bit
TSB	Test and Set Bit

So what exactly do these instructions do? Well, let's examine some of the easy ones first.

PHX, PHY, PLX, PLY

The equivalent of these instructions exist for the accumulator but not for the X and Y registers. One of the more common uses for the stack is to save all the registers prior to going into a routine, so that everything can be restored just prior to exiting.

Ordinarily, to save the A, X, and Y registers, we'd have to do something like this:

ENTRY	PHA	; SAVE A
	TXA	; PUT X IN A
	PHA	; SAVE IT
	TYA	; PUT Y IN A
	PHA	; SAVE IT
WORK	NOP	; YOUR PROGRAM HERE
DONE	PLA	; GET Y
	TAY	; PUT IT BACK
	PLA	; GET X
	TAX	; PUT IT BACK
	PLA	; GET A
EXIT	RTS	

The problem is even further complicated in programs where you need not only to save the registers but to use the accumulator for other purposes also at the same time.

With the new 65C02, this could all be resolved with:

```
ENTRY  PHX      ; SAVE X
        PHY      ; SAVE Y
        PHA      ; SAVE A, BUT LEAVE IT ON TOP

WORK    NOP      ; THE PROGRAM HERE

DONE    PLA      ; GET A
        PLY      ; GET Y
        PLX      ; GET X

EXIT    RTS
```

Addressing Mode	Common Syntax	Hex Coding
-----	-----	-----
Implied only	PHX	DA
	PHY	5A
	PLX	FA
	PLY	7A

BRA (Branch Always)

This is one of those instructions that will thrill writers of relocatable code. In *Assembly Lines: The Book*, it is discussed how to write code that is location independent, and one of the techniques was the use of a forced branch instruction, such as:

```
CLC      ; CLEAR CARRY
BCC LABEL ; ALWAYS
```

Unfortunately, this means we must force some flag of the status register, which may not be convenient at the time. In addition, the process takes up an extra byte on most occasions.

Branch Always alleviates both these problems by always branching to the desired address, subject of course to the usual limitations of plus or minus 128 bytes as the maximum branching distance.

It is worth mentioning in the interest of programming style, that many people indiscriminately use a JMP to go back to the top of a loop when a branch instruction would do the trick, and add one more limitation to the final code in the process. Hopefully, this new branch instruction will encourage people to make their code more location independent.

Addressing Mode	Common Syntax	Hex Coding
-----	-----	-----
Relative only	BRA LABEL1	80 12

STZ (Store Zero)

This instruction is used for zeroing out memory bytes without changing the contents of any of the registers.

Many times it is necessary to set a number of internal program registers to zero before proceeding with the routine. This is especially needed in mathematical routines such as multiplication and division.

Ordinarily, this is done by loading the accumulator with zero, and then storing that value in the appropriate memory locations. This is easy to do when you have to load the A, X or Y registers with zero anyway. The problem is that on occasion, the ONLY reason one of the registers is loaded with zero is because of the need to zero a memory location.

Store Zero allows us to zero out any memory byte without concern to current register contents. The same variety of addressing modes usually available to a STA, STX or STY instruction is not available with a STZ though.

Addressing Mode	Common Syntax	Hex Coding
Absolute	STZ \$1234	9C 34 12
Zero Page	STZ \$12	64 12
Absolute,X	STZ \$1234,X	9E 34 12
Zero Page,X	STZ \$12,X	74 12

SMB, RMB (Set/Reset Memory Bit) [Rockwell chip only]

This pair of nifty instructions will allow you to set or clear a given bit of a byte on the zero page. Previously, this would have required three separate instructions to achieve the same result. For example:

```
LDA MEMORY      ; LOAD VALUE FROM MEMORY
AND #$7F        ; %0111 1111 IS PATTERN
                ; NEEDED TO CLR BIT 7
STA MEMORY      ; PUT IT BACK
```

With the new instruction, we can accomplish the same thing with:

```
RMB7 MEMORY      ; RESET (CLR) BIT 7 OF MEMORY
```

or put it back with:

```
SMB7 MEMORY      ; SET BIT 7 OF MEMORY
```

Two interesting things to note here. The first is that for some reason they use the term "RESET" instead of clear to indicate the zeroing of a given bit.

The second item is that we now have four-character instruction codes (mnemonics). What problems this may cause in some assemblers is open to question, but this new species of instruction seems to have arrived.

These instructions are limited to zero page addressing only.

Addressing Mode	Common Syntax	Hex Coding
-----	-----	-----
Zero Page, Bit 0	SMB0 \$12	87 12
Zero Page, Bit 1	SMB1 \$12	97 12
Zero Page, Bit 2	SMB2 \$12	A7 12
Zero Page, Bit 3	SMB3 \$12	B7 12
Zero Page, Bit 4	SMB4 \$12	C7 12
Zero Page, Bit 5	SMB5 \$12	D7 12
Zero Page, Bit 6	SMB6 \$12	E7 12
Zero Page, Bit 7	SMB7 \$12	F7 12
Zero Page, Bit 0	RMB0 \$12	07 12
Zero Page, Bit 1	RMB1 \$12	17 12
Zero Page, Bit 2	RMB2 \$12	27 12
Zero Page, Bit 3	RMB3 \$12	37 12
Zero Page, Bit 4	RMB4 \$12	47 12
Zero Page, Bit 5	RMB5 \$12	57 12
Zero Page, Bit 6	RMB6 \$12	67 12
Zero Page, Bit 7	RMB7 \$12	77 12

BBS, BBR (Branch on Bit Set/Reset) [Rockwell chip only]

These two new branch instructions make it possible to test any bit of a zero page location, and then branch depending on its condition. This instruction would be very useful for testing flags in programs that need to pack flag-type data into as few bytes as possible. By transferring I/O device registers, etc. to zero page, it is also possible to directly test bits in these registers for status bit conditions.

These instructions are very similar to the Bit Set and Reset instructions just discussed in their appearance and use. The difference is, of course, that we are testing bit status, rather than changing it.

Addressing Mode	Common Syntax	Hex Coding
-----	-----	-----
Zero Page, Bit 0	BBS0 \$12,LABEL	8F 12 FF
Zero Page, Bit 1	BBS1 \$12,LABEL	9F 12 FF
Zero Page, Bit 2	BBS2 \$12,LABEL	AF 12 FF
Zero Page, Bit 3	BBS3 \$12,LABEL	BF 12 FF
Zero Page, Bit 4	BBS4 \$12,LABEL	CF 12 FF
Zero Page, Bit 5	BBS5 \$12,LABEL	DF 12 FF
Zero Page, Bit 6	BBS6 \$12,LABEL	EF 12 FF
Zero Page, Bit 7	BBS7 \$12,LABEL	FF 12 FF
Zero Page, Bit 0	BBR0 \$12,LABEL	0F 12 FF
Zero Page, Bit 1	BBR1 \$12,LABEL	1F 12 FF
Zero Page, Bit 2	BBR2 \$12,LABEL	2F 12 FF
Zero Page, Bit 3	BBR3 \$12,LABEL	3F 12 FF
Zero Page, Bit 4	BBR4 \$12,LABEL	4F 12 FF
Zero Page, Bit 5	BBR5 \$12,LABEL	5F 12 FF
Zero Page, Bit 6	BBR6 \$12,LABEL	6F 12 FF
Zero Page, Bit 7	BBR7 \$12,LABEL	7F 12 FF

One of the first applications that springs to mind is the testing of whether a number is odd or even. Previously, this had to be done with a LSR or ROR instruction, followed by a test of the carry flag, such as:

```
LDA MEMORY      ; LOAD A WITH VALUE
LSR              ; SHIFT BIT 0 INTO CARRY
BCS ODD          ; SET IF ODD
BCC EVEN         ; CLR IF EVEN
```

The equivalent can now be done without effecting the carry flag or the accumulator:

```
BBRO MEMORY,EVEN ; BRANCH IF BIT 0 = 0 = EVEN
BBSO MEMORY,ODD  ; BRANCH IF BIT 0 = 1 = ODD
```

This could be useful also in creating drivers for the new Apple IIe 80 column extended memory board since this card uses one bank of memory or another depending on whether the screen column position is odd or even.

TSB, TRB (Test and Set/Reset Bit)

I've saved the most complex of the new instructions for last. These instructions are rather like a combination of the BIT and AND/ORa instructions of the old 6502.

The instructions seem primarily designed for controlling I/O devices, but may have other interesting applications as things develop.

The action of these two instructions is to use a mask stored in the accumulator to condition a memory location. The mask in the accumulator is unaltered, but the Z flag of the status register is conditioned depending on the memory contents prior to the operation. The Z flag is conditioned based on the result of the accumulator ANDed with the byte in memory like the BIT instruction.

For example, to set both bits 0 and 7 of a memory location, we could use the following set of instructions:

```
LDA #$81          ; %1000 0001 = MASK PATTERN
TSB MEM1          ; SET BITS 0,7 OF MEMORY
BNE PRSET         ; ONE OF THESE WAS 'ON' ALREADY
BEQ PRCLR         ; NONE OF THESE WERE 'ON' ALREADY
```

This would clear the bits:

```
LDA #$81          ; %1000 0001 = MASK PATTERN
TRB MEM2          ; CLR BIT 0,7 OF MEMORY
BNE PRSET         ; ONE OF THESE WAS 'ON' ALREADY
BEQ PRCLR         ; NONE OF THESE WERE 'ON' ALREADY
```


The addressing modes allowed for these instructions are as follows:

Addressing Mode	Common Syntax	Hex Coding
-----	-----	-----
Absolute	TRB \$1234	1C 34 12
Zero Page	TRB \$12	14 12
Absolute	TSB \$1234	0C 34 12
Zero Page	TSB \$12	04 12

OTHER DIFFERENCES

There are a number of other differences in the chips, most notably the power consumption. The power use of the 65C02 is one-tenth that of the 6502 so the chip runs considerably cooler, not to mention the possibilities opened for portable computers, terminals, etc.

One point of interest is that the old 6502 bug of the indirect jump problem has been fixed. If you're not aware of it, the 6502 has a well-documented problem with indirect jumps that use a pair of bytes that straddle a page boundary.

For example, consider these two instructions:

Instruction	Pointers Wanted	Pointers Used
-----	-----	-----
JMP (\$380)	\$380,381	\$380,381
JMP (\$3FF)	\$3FF,400	\$3FF,300

Notice that in the last instance, the pointers used are NOT those anticipated. This is because the high byte of the pointer address does not get properly incremented on the standard 6502.

This problem has been fixed on the 65C02. The only possible problem here is "clever" protection schemes that may have used this bug to throw off people trying to decode the protection method. Otherwise, this should not present any problems to existing software.

Are there any problems to be anticipated? In theory, no. The new 65C02 is pin-for-pin compatible with the old one, and also upward compatible in terms of software. ANY software for the Apple, PET, Atari or other 6502 based microcomputers SHOULD work without problems with the new chip. Unfortunately, yes.

The first big problem concerns internal microprocessor timing on the Apple II and II+ computers. From the available information it seems that the Apple II and II+ do not handle the CPU clock cycles in the same way that the IIe does.

On the surface, the 65C02 should directly replace the 6502. However, because the 65C02 is a faster chip, data is not available for as long, and bits can get lost. What this means for now is that the 65C02 can only be used on the Apple IIe and Apple III machines. None of the manufacturers at this time produce a chip that works in an Apple II or II+.

It can be expected though that revisions will be made in the near future that will allow the 65C02 to be implemented on the older machines.

There is also the possibility of problems with some existing software. A very small percentage of software may be using (it's not really known) undocumented bugs or "features" of the old 6502 chip, and these might not function as anticipated.

For example, a reasonable question might be, "Where did all the new opcodes come from? After all, wasn't the chip 'full'?" To answer this, consider how the instructions we use now are structured. The 6502 operates by scanning memory and performing specific operations depending on the values that it finds in each memory location. You would expect a total of 256 possible opcodes then. As it happens, all 256 possible values are not used. It is this group of "unused" opcodes that allows for the "new" instructions, and also creates the possibility for a small percentage of difficulties with existing programs.

Although rarely documented, the previously "unused" values will cause certain things to happen, much the same way that a legal value would. For instance, the code \$FF on a 6502 is labeled as an alternate NOP. This is one of the codes that has been converted to a new function in the 65C02, namely BBS7 (Branch on Bit 7 Set).

There are other unused codes though that have combination effects, usually of little use, such as loading the accumulator and decrementing a register at the same time. Their main application is similar to the indirect jump problem, i.e. creating code that cannot be casually interpreted. If these instructions have been used in existing software though, problems could arise with the 65C02.

With such difficulties then, why bother to substitute the new 65C02 into an existing Apple? The answers are varied.

First of all, the 65C02 is likely to appear in upcoming releases of existing computers (maybe a new release of the IIe even?) and as such, you can experiment now with the newest version of this versatile device.

Second, there are likely to be specific applications where the advantages of the chip will make it worth supplying with the software, since the disadvantages are practically non-existent for Apple IIe and Apple III. Code re-written to take advantage of the new instructions can be expected to be 10-15% smaller and run proportionally faster. In certain applications, even greater improvements could be expected.

At this writing, the Rockwell chip seems to have the largest set of instructions of the three varieties available. The GTE and NCR chips lack the bit manipulation instructions (BBS, BBR, SMB, and RMB), but are otherwise identical.

As far as assemblers that support the instructions, the current version of MERLIN supports all the new opcodes in both the assembly and SOURCEROR portions of the product. The S-C Assembler and Hayden's ORCA/M assembler will also be supplying updates. Other assemblers that support MACRO capabilities though should be able to be used immediately simply by defining the proper hex codes.

If you have any difficulties with your 65C02, please contact Southwestern Data Systems at 619-562-3670.

Rewrite

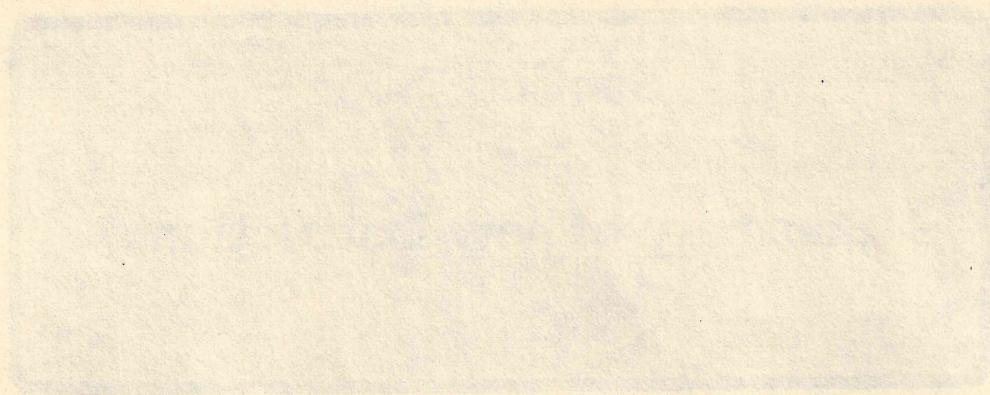
...for an ...
...of ...
...the ...
...and ...
...the ...
...the ...
...the ...

...the ...
...the ...
...the ...

Revised

Section Two

Summary of new Instructions



SECTION TWO

SUMMARY OF NEW INSTRUCTIONS

THE FOLLOWING 6502 INSTRUCTIONS GET NEW ADDRESSING MODES

ADC
AND
BIT
CMP
DEC
EOR
INC
JMP
LDA
ORA
STA

The following instructions get ZERO PAGE INDIRECT ADDRESSING

ADC
AND
CMP
EOR
LDA
ORA
STA

ACCUMULATOR ADDRESSING

DEC
INC

ABSOLUTE Pre-Indexed Indirect Addressing

JMP (addr16,X) Usefull for jump tables.

The BIT instruction has the following additional addressing modes.

BIT imm.
BIT ZP,X
BIT ABS,X

The following new instructions have been added.

BBR	Branch on Bit Reset	[ROCKWELL CHIP ONLY]
BBS	Branch on Bit Set	[ROCKWELL CHIP ONLY]
BRA	Branch Always	
PHX	Push X Register on Stack	
PHY	Push Y Register on Stack	
PLX	Pull X Register from Stack	
PLY	Pull Y Register from Stack	
RMB	Reset Memory Bit	[ROCKWELL CHIP ONLY]
SMB	Set Memory Bit	[ROCKWELL CHIP ONLY]
STZ	Store Zero	
TRB	Test and Reset Bits	
TSB	Test and Set Bits	

-> BRA Branch always

MODE	SYNTAX	OPCODE	BYTES	CYCLES
relative	BRA	\$80	2	3

DESCRIPTION: BRA executes an unconditional branch. This is a relative branch and as such is limited to +129/-126 offset from the address of the instruction.

USES: This additional branch is useful in writing position independent code. It executes in the same number of cycles (3) as an absolute JMP, but requires only two bytes. This results in tighter code if the relative offset is close enough.

-> PHX Push X Register on Stack

MODE	SYNTAX	OPCODE	BYTES	CYCLES
Implied	PHX	\$DA	1	3

DESCRIPTION: Save the contents of the X register on the stack.

-> PHY Push Y Register on Stack

MODE	SYNTAX	OPCODE	BYTES	CYCLES
Implied	PHY	\$5A	1	3

DESCRIPTION: Save the contents of the Y register on the stack.

-> PLX Pull X Register from Stack

MODE	SYNTAX	OPCODE	BYTES	CYCLES
Implied	PLX	\$FA	1	4

DESCRIPTION: Load the X register from the stack.

-> PLY Pull Y Register from Stack

MODE	SYNTAX	OPCODE	BYTES	CYCLES
Implied	PLY	\$7A	1	4

NOTE: PLX and PLY affect the processor status register as PLA.

-> RMB Reset Memory Bit [Rockwell Chip Only]

MODE	MNEMONIC	OPCODE	BYTES	CYCLES
Zero Page	RMB0	\$07	2	5
Zero Page	RMB1	\$17	2	5
Zero Page	RMB2	\$27	2	5
Zero Page	RMB3	\$37	2	5
Zero Page	RMB4	\$47	2	5
Zero Page	RMB5	\$57	2	5
Zero Page	RMB6	\$67	2	5
Zero Page	RMB7	\$77	2	5

msb							lsb									
	B7		B6		B5		B4		B3		B2		B1		B0	

[Rockwell Chip Only]

MODE	MNEMONIC	OPCODE	BYTES	CYCLES
Zero Page	SMB0	\$87	2	5
Zero Page	SMB1	\$97	2	5
Zero Page	SMB2	\$A7	2	5
Zero Page	SMB3	\$B7	2	5
Zero Page	SMB4	\$C7	2	5
Zero Page	SMB5	\$D7	2	5
Zero Page	SMB6	\$E7	2	5
Zero Page	SMB7	\$F7	2	5

-> STZ Store Zero

MODE	SYNTAX	OPCODE	BYTES	CYCLES
Absolute	STZ addr16	\$9C	3	4
Zero Page	STZ addr8	\$64	2	3
Zero Page,X	STZ addr8,X	\$74	2	4
Abs,X	STZ addr16,X	\$9E	3	5

DESCRIPTION: Store a zero in the specified memory location. This instruction does not affect the status register.

[illegible]

-> TRB Test and Reset Bits

MODE	SYNTAX	OPCODE	BYTES	CYCLES
Absolute	TRB addr16	\$1C	3	6
Zero Page	TRB addr8	\$14	2	5

OPERATION: $\bar{A}$ AND M -> M

DESCRIPTION: Reset the the bits of the specified memory location based on the mask in the accumulator. The accumulator is left unaltered. The Z flag is conditioned based on the result A AND M before the operation takes place (ie. BIT instruction).

N	V	-	B	D	I	Z	C	ACC	X	Y	MEM
						X					X

-> TSB Test and Set Bits

MODE	SYNTAX	OPCODE	BYTES	CYCLES
Absolute	TSB addr16	\$0C	3	6
Zero Page	TSB addr8	\$04	2	5

OPERATION: A OR M -> M

DESCRIPTION: Set the the bits of the specified memory location based on the mask in the accumulator. The accumulator is left unaltered. The Z flag is conditioned based on the result A AND M before the operation takes place (ie. BIT instruction).

N	V	-	B	D	I	Z	C	ACC	X	Y	MEM
						X					X

-> BBRn Branch on Bit n Reset.

[Rockwell Chip Only]

MODE	SYNTAX	OPCODE	BYTES	CYCLES
RELATIVE	BBR0 addr8,offset	\$0F	3	5+n
RELATIVE	BBR1 addr8,offset	\$1F	3	5+n
RELATIVE	BBR2 addr8,offset	\$2F	3	5+n
RELATIVE	BBR3 addr8,offset	\$3F	3	5+n
RELATIVE	BBR4 addr8,offset	\$4F	3	5+n
RELATIVE	BBR5 addr8,offset	\$5F	3	5+n
RELATIVE	BBR6 addr8,offset	\$6F	3	5+n
RELATIVE	BBR7 addr8,offset	\$7F	3	5+n

n = 0 if branch does not occur

n = 1 if branch occurs to same page

n = 2 if branch occurs to different page

Description: This instruction causes a Branch to take place if the selected bit of the specified zero page location is reset (=0). No flags are affected.

-> BBSn Branch on Bit n Set.

[Rockwell Chip Only]

MODE	SYNTAX	OPCODE	BYTES	CYCLES
RELATIVE	BBS0 addr8,offset	\$8F	3	5+n
RELATIVE	BBS1 addr8,offset	\$9F	3	5+n
RELATIVE	BBS2 addr8,offset	\$AF	3	5+n
RELATIVE	BBS3 addr8,offset	\$BF	3	5+n
RELATIVE	BBS4 addr8,offset	\$CF	3	5+n
RELATIVE	BBS5 addr8,offset	\$DF	3	5+n
RELATIVE	BBS6 addr8,offset	\$EF	3	5+n
RELATIVE	BBS7 addr8,offset	\$FF	3	5+n

n = 0 if branch does not occur

n = 1 if branch occurs to same page

n = 2 if branch occurs to different page

Description: This instruction causes a Branch to take place if the selected bit of the specified zero page location is set (=1). No flags are affected.

(Revised) Chip Data

-> Error Search on Bit 0 Error

MODE	SYNTAX	ADDRESS	EXTEND	STATUS
RELATIVE	8000	0000	0	0
RELATIVE	8000	0000	0	0
RELATIVE	8000	0000	0	0
RELATIVE	8000	0000	0	0
RELATIVE	8000	0000	0	0
RELATIVE	8000	0000	0	0
RELATIVE	8000	0000	0	0
RELATIVE	8000	0000	0	0
RELATIVE	8000	0000	0	0
RELATIVE	8000	0000	0	0

n = 0 if branch does not occur
n = 1 if branch occurs to same page
n = 2 if branch occurs to different page

Description: This instruction makes a branch to the place if
the selected bit of the specified byte location is zero.
(-0). No flags are affected.

(Revised) Chip Data

-> Error Search on Bit 1 Error

MODE	SYNTAX	ADDRESS	EXTEND	STATUS
RELATIVE	8000	0000	0	0
RELATIVE	8000	0000	0	0
RELATIVE	8000	0000	0	0
RELATIVE	8000	0000	0	0
RELATIVE	8000	0000	0	0
RELATIVE	8000	0000	0	0
RELATIVE	8000	0000	0	0
RELATIVE	8000	0000	0	0
RELATIVE	8000	0000	0	0
RELATIVE	8000	0000	0	0

n = 0 if branch does not occur
n = 1 if branch occurs to same page
n = 2 if branch occurs to different page

Description: This instruction makes a branch to the place if
the selected bit of the specified byte location is not zero.
No flags are affected.

Section Three

Quick Opcode Reference Table

This section is intended for those who
program in assembly language
and need a chart of instructions to refer to.

The chart includes:

Each instruction, each legal addressing mode
of that instruction, what the opcode is
(in hexadecimal), the number of bytes the instruction
requires, the number of cycles the instruction takes
to execute, and the flags in the p-register
that the instruction affects.

Name Description	Operation	Addressing Mode	Assembly Language Form
ADC (*d) Add memory to accumulator with carry	$A+M+C \rightarrow A, C$	Immediate Zero Page Zero Page,X Absolute Absolute,X Absolute,Y (Indirect,X) (Indirect),Y (Indirect) (*n)	ADC #Oper ADC Oper ADC Oper,X ADC Oper ADC Oper,X ADC Oper,Y ADC (Oper,X) ADC (Oper),Y ADC (Oper)
AND "And" memory with accumulator	$A \text{ and } M \rightarrow A$	Immediate Zero Page Zero Page,X Absolute Absolute,X Absolute,Y (Indirect,X) (Indirect),Y (Indirect) (*n)	AND #Oper AND Oper AND Oper,X AND Oper AND Oper,X AND Oper,Y AND (Oper,X) AND (Oper),Y AND (Oper)
ASL Shift left one bit (memory or accumulator)	$C \leftarrow 76543210 \leftarrow 0$	Accumulator Zero Page Zero Page,X Absolute Absolute,X	ASL {A} ASL Oper ASL Oper,X ASL Oper ASL Oper,X
BBRx (*n) Branch on bit reset	Branch if the specified bit in the zero page location = 0	Z Page/Relative	BBRx Oper,Oper

Name	Hex Opcode	Number of bytes	Number of cycles	"P" Status Reg.					
				N	Z	C	I	D	V
ADC Immediate	69	2	2	*	*	*			*
ADC Zero Page	65	2	3						
ADC Zero Page,X	75	2	4						
ADC Absolute	6D	3	4						
ADC Absolute,X	7D	3	4 (*c)						
ADC Absolute,Y	79	3	4 (*c)						
ADC (Indirect,X)	61	2	6						
ADC (Indirect),Y	71	2	5 (*c)						
ADC (Indirect)	72	2	5						
AND Immediate	29	2	2	*				*	
AND Zero Page	25	2	3						
AND Zero Page,X	35	2	4						
AND Absolute	2D	3	4						
AND Absolute,X	3D	3	4 (*c)						
AND Absolute,Y	39	3	4 (*c)						
AND (Indirect,X)	21	2	6						
AND (Indirect),Y	31	2	5 (*c)						
AND (Indirect)	32	2	5						
ASL Accumulator	0A	1	2	*	*	*			
ASL Zero Page	06	2	5						
ASL Zero Page,X	16	2	6						
ASL Absolute	0E	3	6						
ASL Absolute,X	1E	3	7						
BBR0 Loc,Offset	0F	3	5 (*b)						
BBR1 Loc,Offset	1F								
BBR2 Loc,Offset	2F								
BBR3 Loc,Offset	3F								
BBR4 Loc,Offset	4F								
BBR5 Loc,Offset	5F								
BBR6 Loc,Offset	6F								
BBR7 Loc,Offset	7F								

Name Description	Operation	Addressing Mode	Assembly Language Form
BBSx (*n) Branch on bit set	Branch if the specified bit in the byte on the zero page = 1	Z Page/Relative	BBSx Oper,Oper
BCC	Branch if carry=0	Relative	BCC Oper
BCS	Branch if carry=1	Relative	BCS Oper
BEQ	Branch if zero=1	Relative	BEQ Oper
BIT Tests bits in memory with accumulator	A and M. M7 -> N M6 -> V	Immediate (*n*a) Zero Page Zero Page,X (*n) Absolute Absolute,X (*n)	BIT #Oper BIT Oper BIT Oper,X BIT Oper BIT Oper,X
BMI	Branch if N=1	Relative	BMI Oper
BNE	Branch if Z=0	Relative	BNE Oper
BPL	Branch if N=0	Relative	BPL Oper
BRA (*n)	Branch always	Relative	BRA Oper
BRK	Forced Interrupt	Implied	BRK
BVC	Branch if V=0	Relative	BVC Oper
BVS	Branch if V=1	Relative	BVS Oper
CLC clear carry	0 -> carry	Implied	CLC
CLD clear decimal	0 -> decimal	Implied	CLD
CLI clear interrupt	0 -> interrupt	Implied	CLI
CLV clear overflow	0 -> overflow	Implied	CLV
CMP compare memory to accumulator	A - M	Immediate Zero Page Zero Page,X Absolute Absolute,X (Indirect,X) (Indirect),Y	CMP #Oper CMP Oper CMP Oper,X CMP Oper CMP Oper,X CMP (Oper,X) CMP (Oper),Y

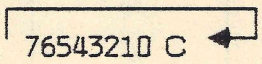
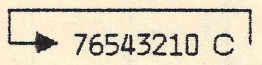
Name	Hex Opcode	Number of bytes	Number of cycles	"P" Status Reg.					
				N	Z	C	I	D	V
BBS0 Loc,Offset	8F	3	5 (*b)						
BBS1 Loc,Offset	9F								
BBS2 Loc,Offset	AF								
BBS3 Loc,Offset	BF								
BBS4 Loc,Offset	CF								
BBS5 Loc,Offset	DF								
BBS6 Loc,Offset	EF								
BBS7 Loc,Offset	FF								
BCC Offset	90	2	2 (*b)						
BCS Offset	B0	2	2 (*b)						
BEQ Offset	F0	2	2 (*b)						
BIT Immediate	89	2	2	*	*				
BIT Zero Page	24	2	3	*	*				*
BIT Zero Page,X	34	2	4	*	*				*
BIT Absolute	2C	3	4	*	*				*
BIT Absolute,X	3C	3	4 (*c)	*	*				*
BMI Offset	30	2	2 (*b)						
BNE Offset	D0	2	2 (*b)						
BPL Offset	10	2	2 (*b)						
BRA Offset	80	2	2 (*b)						
BRK	00	1	7						
BVC Offset	50	2	2 (*b)						
BVS Offset	70	2	2 (*b)						
CLC	18	1	2			0			
CLD	D8	1	2					0	
CLI	58	1	2				0		
CLV	B8	1	2						0
CMP Immediate	C9	2	2	*	*	*			
CMP Zero Page	C5	2	3						
CMP Zero Page,X	D5	2	4						
CMP Absolute	CD	3	4						
CMP Absolute,X	DD	3	4 (*c)						
CMP Absolute,Y	D9	3	4 (*c)						
CMP (Indirect,X)	C1	2	6						

Name Description	Operation	Addressing Mode	Assembly Language Form
CMP (continued)		(Indirect),Y (Indirect) (*n)	CMP (Oper),Y CMP (Oper)
CPX Compare memory to X register	X - M	Immediate Zero Page Absolute	CPX #Oper CPX Oper CPX Oper
CPY Compare memory to Y register	Y - M	Immediate Zero Page Absolute	CPY #Oper CPY Oper CPY Oper
DEC Decrement memory by one	M - 1 -> M	Zero Page Zero Page,X Absolute Absolute,X Accumulator (*n)	DEC Oper DEC Oper,X DEC Oper DEC Oper,X DEC
DEX Decrement X reg	X - 1 -> X	Implied	DEX
DEY Decrement Y reg	Y - 1 -> Y	Implied	DEY
EOR "Exclusive-or" memory with accumulator	A xor M -> A	Immediate Zero Page Zero Page,X Absolute Absolute,X Absolute,Y (Indirect,X) (Indirect),Y (Indirect) (*n)	EOR #Oper EOR Oper EOR Oper,X EOR Oper EOR Oper,X EOR Oper,Y EOR (Oper,X) EOR (Oper),Y EOR (Oper)
INC Increment memory (or accumulator) by one	M + 1 -> M	Zero Page Zero Page,X Absolute Absolute,X Accumulator	INC Oper INC Oper,X INC Oper INC Oper,X INC
INX Increment X reg	X + 1 -> X	Implied	INX
INY Increment Y reg	Y + 1 -> Y	Implied	INY
JMP Jump to new address	(PC+1)->PCL (PC+2)->PCH	Absolute (Indirect) (AbsIndirect,X)*n	JMP Oper JMP (Oper) JMP (Oper,X)

Name	Hex Opcode	Number of bytes	Number of cycles	"P" Status Reg.					
				N	Z	C	I	D	V
CMP (Indirect),Y	D1	2	5 (*c)	*	*	*			
CMP (Indirect)	D2	2	5						
CPX Immediate	E0	2	2	*	*	*			
CPX Zero Page	E4	2	3						
CPX Absolute	EC	3	4						
CPY Immediate	C0	2	2	*	*	*			
CPY Zero Page	C4	2	3						
CPY Absolute	CC	3	4						
DEC Zero Page	C6	2	5	*	*				
DEC Zero Page,X	D6	2	6						
DEC Absolute	CE	3	6						
DEC Absolute,X	DE	3	7						
DEC {accumulator}	3A	1	2						
DEX	CA	1	2	*	*				
DEY	88	1	2	*	*				
EOR Immediate	49	2	2	*	*				
EOR Zero Page	45	2	3						
EOR Zero Page,X	55	2	4						
EOR Absolute	4D	3	4						
EOR Absolute,X	5D	3	4 (*c)						
EOR Absolute,Y	59	3	4 (*c)						
EOR (Indirect,X)	41	2	6						
EOR (Indirect),Y	51	2	5 (*c)						
EOR (Indirect)	52	2	5						
INC Zero Page	E6	2	5	*	*				
INC Zero Page,X	F6	2	6						
INC Absolute	EE	3	6						
INC Absolute,X	FE	3	7						
INC {accumulator}	1A	1	2						
INX	E8	1	2	*	*				
INY	C8	1	2	*	*				
JMP Absolute	4C	3	3						
JMP (Indirect)	6C	3	5						
JMP (AbsIndirect,X)	7C	3	6						

Name Description	Operation	Addressing Mode	Assembly Language Form
JSR Jump to subroutine	PC+2->stack, etc	Absolute	JSR Oper
LDA Load accumulator	M -> A	Immediate Zero Page Zero Page,X Absolute Absolute,X Absolute,Y (Indirect,X) (Indirect),Y (Indirect) (*n)	LDA #Oper LDA Oper LDA Oper,X LDA Oper LDA Oper,X LDA Oper,Y LDA (Oper,X) LDA (Oper),Y LDA (Oper)
LDX Load X register with memory	M -> X	Immediate Zero Page Zero Page,Y Absolute Absolute,Y	LDX #Oper LDX Oper LDX Oper,Y LDX Oper LDX Oper,Y
LDY Load Y register with memory	M -> Y	Immediate Zero Page Zero Page,X Absolute Absolute,X	LDY #Oper LDY Oper LDY Oper,X LDY Oper LDY Oper,X
LSR Logical shift right	shift right one bit, 0->M7	Zero Page Zero Page,X Absolute Absolute,X Accumulator	LSR Oper LSR Oper,X LSR Oper LSR Oper,X LSR
ORA "Or" accumulator with memory	A or M -> M	Immediate Zero Page Zero Page,X Absolute Absolute,X Absolute,Y (Indirect,X) (Indirect),Y (Indirect) (*n)	ORA #Oper ORA Oper ORA Oper,X ORA Oper ORA Oper,X ORA Oper,Y ORA (Oper,X) ORA (Oper),Y ORA (Oper)

Name	Hex Opcode	Number of bytes	Number of cycles	"P" Status Reg.					
				N	Z	C	I	D	V
JSR Absolute	20	3	6						
LDA Immediate	A9	2	2	*	*				
LDA Zero Page	A5	2	3						
LDA Zero Page,X	B5	2	4						
LDA Absolute	AD	3	4						
LDA Absolute,X	BD	3	4 (*c)						
LDA Absolute,Y	B9	3	4 (*c)						
LDA (Indirect,X)	A1	2	6						
LDA (Indirect),Y	B1	2	5 (*c)						
LDA (Indirect)	B2	2	5						
LDX Immediate	A2	2	2	*	*				
LDX Zero Page	A6	2	3						
LDX Zero Page,Y	B6	2	4						
LDX Absolute	AE	3	4						
LDX Absolute,Y	BE	3	4 (*c)						
LDY Immediate	A0	2	2	*	*				
LDY Zero Page	A4	2	3						
LDY Zero Page,X	B4	2	4						
LDY Absolute	AC	3	4						
LDY Absolute,X	BC	3	4 (*c)						
LSR Zero Page	46	2	5	0	*	*			
LSR Zero Page,X	56	2	6						
LSR Absolute	4E	3	6						
LSR Absolute,X	5E	3	7						
LSR {accumulator}	4A	1	2						
ORA Immediate	09	2	2	*	*				
ORA Zero Page	05	2	3						
ORA Zero Page,X	15	2	4						
ORA Absolute	0D	3	4						
ORA Absolute,X	1D	3	4 (*c)						
ORA Absolute,Y	19	3	4 (*c)						
ORA (Indirect,X)	01	2	6						
ORA (Indirect),Y	11	2	5 (*c)						
ORA (Indirect)	12	2	5						

Name Description	Operation	Addressing Mode	Assembly Language Form
PHA	A → stack	Implied	PHA
PHP	P → stack	Implied	PHP
PHX (*n)	X → stack	Implied	PHX
PHY (*n)	Y → stack	Implied	PHY
PLA	stack → A	Implied	PLA
PLP	stack → P	Implied	PLP
PLX (*n)	stack → X	Implied	PLX
PLY (*n)	stack → Y	Implied	PLY
RMBx Reset (clear) memory bit in Zero Page byte	Mx → 0	Zero Page	RMBx Oper
ROL Rotate one bit left (memory or accumulator)		Accumulator Zero Page Zero Page,X Absolute Absolute,X	ROL ROL Oper ROL Oper,X ROL Oper ROL Oper,X
ROR Rotate one bit right (memory or accumulator)		Accumulator Zero Page Zero Page,X Absolute Absolute,X	ROR ROR Oper ROR Oper,X ROR Oper ROR Oper
RTI	Interrupt return	Implied	RTI
RTS	Subroutine return	Implied	RTS
SBC (*d) Subtract memory from accumulator with borrow (taken from the carry bit)	A - M - $\overline{C}$ → A	Immediate Zero Page Zero Page,X Absolute Absolute,X Absolute,Y	SBC #Oper SBC Oper SBC Oper,X SBC Oper SBC Oper,X SBC Oper,Y

Name	Hex Opcode	Number of bytes	Number of cycles	"P" Status Reg.					
				N	Z	C	I	D	V
PHA	48	1	3						
PHP	08	1	3						
PHX	DA	1	3						
PHY	5A	1	3						
PLA	68	1	4	*	*				
PLP	28	1	4	*	*	*	*	*	*
PLX	FA	1	4	*	*				
PLY	7A	1	4	*	*				
RMB0 Zero Page	07	2	5						
RMB1 Zero Page	17								
RMB2 Zero Page	27								
RMB3 Zero Page	37								
RMB4 Zero Page	47								
RMB5 Zero Page	57								
RMB6 Zero Page	67								
RMB7 Zero Page	77								
ROL {accumulator}	2A	1	2	*	*	*			
ROL Zero Page	26	2	5						
ROL Zero Page,X	36	2	6						
ROL Absolute	2E	3	6						
ROL Absolute,X	3E	3	7						
ROR {accumulator}	6A	1	2	*	*	*			
ROR Zero Page	66	2	5						
ROR Zero Page,X	76	2	6						
ROR Absolute	6E	3	6						
ROR Absolute,X	7E	3	7						
RTI	40	1	6	*	*	*	*	*	*
RTS	60	1	6						
SBC Immediate	E9	2	2	*	*	*			*
SBC Zero Page	E5	2	3						
SBC Zero Page,X	F5	2	4						
SBC Absolute	ED	3	4						
SBC Absolute,X	FD	3	4 (*c)						
SBC Absolute,Y	F9	3	4 (*c)						

Name Description	Operation	Addressing Mode	Assembly Language Form
SBC (cont.)		(Indirect,X) (Indirect),Y (Indirect) (*n)	SBC (Oper,X) SBC (Oper),Y SBC (Oper)
SEC set carry bit	carry = 1	Implied	SEC
SED set BCD mode	decimal = 1	Implied	SED
SEI set interrupt flag	interrupt = 1	Implied	SEI
SMBx (*n) Set specified bit in byte on zero page	Mx = 1	Zero Page	SMBx Oper
STA Store accumulator in memory	A → M	Zero Page Zero Page,X Absolute Absolute,X Absolute,Y (Indirect,X) (Indirect),Y (Indirect) (*n)	STA Oper STA Oper,X STA Oper STA Oper,X STA Oper,Y STA (Indirect,X) STA (Indirect),Y STA (Indirect)
STX Store X register in memory	X → M	Zero Page Zero Page,Y Absolute	STX Oper STX Oper,Y STX Oper
STY Store Y register in memory	Y → M	Zero Page Zero Page,X Absolute	STY Oper STY Oper,X STY Oper
STZ (*n) Store 0 in memory	0 → M	Zero Page Zero Page,X Absolute Absolute,X	STZ Oper STZ Oper,X STZ Oper STZ Oper,X
TAX Transfer A to X	A → X	Implied	TAX
TAY Transfer A to Y	A → Y	Implied	TAY

Name	Hex Opcode	Number of bytes	Number of cycles	"P" Status Reg.					
				N	Z	C	I	D	V
SBC (Indirect,X)	E1	2	6	*	*	*			*
SBC (Indirect),Y	F1	2	5 (*C)						
SBC (Indirect)	F2	2	5						
SEC	38	1	2			1			
SED	F8	1	2					1	
SEI	78	1	2				1		
SMB0 Zero Page	87	2	5						
SMB1 Zero Page	97								
SMB2 Zero Page	A7								
SMB3 Zero Page	B7								
SMB4 Zero Page	C7								
SMB5 Zero Page	D7								
SMB6 Zero Page	E7								
SMB7 Zero Page	F7								
STA Zero Page	85	2	3						
STA Zero Page,X	95	2	4						
STA Absolute	8D	3	4						
STA Absolute,X	9D	3	5						
STA Absolute,Y	99	3	5						
STA (Indirect,X)	81	2	6						
STA (Indirect),Y	91	2	6						
STA (Indirect)	92	2	5						
STX Zero Page	86	2	3						
STX Zero Page,Y	96	2	4						
STX Absolute	8E	3	4						
STY Zero Page	84	2	3						
STY Zero Page,X	94	2	4						
STY Absolute	8C	3	4						
STZ Zero Page	64	2	3						
STZ Zero Page,X	74	2	4						
STZ Absolute	9C	3	4						
STZ Absolute,X	9E	3	5						
TAX	AA	1	2	*	*				
TAY	A8	1	2	*	*				

Name Description	Operation	Addressing Mode	Assembly Language Form
TRB test and reset bits by accum. mask	$\bar{A}$ and $M \rightarrow M$	Zero Page Absolute	Zero Page Absolute
TSB test and set bits by accumulator mask	A or $M \rightarrow M$	Zero Page Absolute	Zero Page Absolute
TSX transfer S to X	$S \rightarrow X$	Implied	TSX
TXA transfer X to A	$X \rightarrow A$	Implied	TXA
TXS transfer X to S	$X \rightarrow S$	Implied	TXS
TYA transfer Y to A	$Y \rightarrow A$	Implied	TYA
(and last but not least...) NOP	No operation	Implied	NOP

(*a)-extra notes on BIT #Immediate-

BIT #Immediate doesn't transfer bits 6 and 7 of the immediate operand to the N and V flags. The N flag is conditioned by the result of the AND that conditions the Z flag.

(*b)-add one cycle if branch succeeds, add two if crossing page boundary

(*c)-add one cycle if crossing page boundary

(*d)-add one cycle if in decimal (BCD) mode

(*n)-new instruction (or old instruction in new addressing mode)

(NOTE: This section of documentation done on the **LISA** computer!)

Name	Hex Opcode	Number of bytes	Number of cycles	"P" Status Reg.					
				N	Z	C	I	D	V
TRB Zero Page	14	2	5		*				
TRB Absolute	1C	3	6						
TSB Zero Page	04	2	5		*				
TSB Absolute	0C	3	6						
TSX	BA	1	2	*	*				
TXA	8A	1	2	*	*				
TXS	9A	1	2						
TYA	98	1	2	*	*				
NOP	EA	1	2						

Date		Description		Amount	
1901	Jan 1	Balance		100.00	
1901	Jan 15	Received from A. B.		50.00	
1901	Feb 1	Received from C. D.		25.00	
1901	Mar 1	Received from E. F.		75.00	
1901	Apr 1	Received from G. H.		100.00	
1901	May 1	Received from I. J.		150.00	
1901	Jun 1	Received from K. L.		200.00	
1901	Jul 1	Received from M. N.		250.00	
1901	Aug 1	Received from O. P.		300.00	
1901	Sep 1	Received from Q. R.		350.00	
1901	Oct 1	Received from S. T.		400.00	
1901	Nov 1	Received from U. V.		450.00	
1901	Dec 1	Received from W. X.		500.00	
1901	Dec 31	Total		2500.00	

**Rockwell**

R65C00 CMOS Microcomputer System DATA SHEET

R65C00 MICROPROCESSORS (CPU)

DESCRIPTION

The 8-bit R65C00 microcomputer system is produced with CMOS Silicon Gate technology. Advanced system architecture enhances its performance speeds; a family of software-compatible microprocessor (CPU) devices (described below) enhances system cost-effectiveness. Rockwell also provides memory and microcomputer systems, as well as low-cost design aids and documentation.

R65C00 MICROPROCESSOR (CPU) CONCEPT

Three CPU devices are available. All are software-compatible and provide addressable memory, interrupt input, and on-chip clock oscillators and drivers options. All are bus-compatible with the NMOS R6500 family devices.

The family includes two microprocessors with on-board clock oscillators and drivers and one microprocessor driven by external clocks. The on-chip clock versions are aimed at high performance, low cost applications where single phase inputs, crystal or RC inputs provide the time base. The slave processor version is geared for multiprocessor system applications where maximum timing control is mandatory. All R65C00 microprocessors are available in a variety of packaging (ceramic and plastic), operating frequency (2 MHz, 3 MHz and 4 MHz), and temperature (commercial and industrial) versions.

MEMBERS OF THE R65C00 MICROPROCESSOR (CPU) FAMILY

Microprocessors with Internal Clock Generator:

Model	Addressable Memory
R65C02	64K Bytes
R65C102	64K Bytes

Microprocessors with External Clock Input:

Model	Addressable Memory
R65C112	64K Bytes

FEATURES

- CMOS silicon gate technology
- Low Power (4mA/MHz)
- Downward software compatible with R6502
 - Twelve additional instructions
 - Two new addressing modes
- Single 5V $\pm 20\%$ power supply
- Eight bit parallel processing
- Decimal and binary arithmetic
- True indexing capability
- Programmable stack pointer
- Interrupt capability
- Non-maskable interrupt
- Use with any type of speed memory
- Eight-bit Bidirectional Data Bus
- Addressable memory range of up to 64K bytes
- "Ready" input
- Direct Memory Access capability
- Memory Lock Output
- 2MHz, 3MHz, and 4MHz versions
- Choice of external or on-chip clocks
 - External single clock input
 - Direct Crystal Input ($\div 4$)
- Commercial and industrial temperature versions
- Pipeline architecture
- Slave Processor Version (R65C112)

ORDERING INFORMATION

ORDER NUMBER:

R65C102
R65C02
R65C112

Temp Range
No Suffix = 0°C to +70°C
E = -40°C to +85°C

Package C = Ceramic
P = Plastic

Frequency Range
A = 2 MHz
B = 3 MHz
C = 4 MHz

R65C00 MICROPROCESSORS (CPU)

SIGNAL DESCRIPTION

Clocks (ϕ_0 , ϕ_1 , ϕ_2 , ϕ_4)

The R65C112 requires an external ϕ_2 clock.

The R65C02 requires an external ϕ_0 clock.

The R65C102 clocks may be generated externally or internally with a crystal across XTLI and XTLO.

ϕ_0 —TTL input clock to the R65C02

ϕ_4 —Quadrature output clock from the R65C102. The address is valid at the rising edge of ϕ_4 .

When the input clock is stopped the CPU is in the standby mode.

Address Bus (A0-A15)

These outputs are TTL compatible and capable of driving one standard TTL load and 130 pF.

Data Bus (D0-D7)

The data bus uses eight pins. This is a bidirectional bus, transferring data to and from the device and peripherals. The outputs are tri-state buffers capable of driving one standard TTL load and 130 pF.

Ready (RDY)

This input signal allows the user to halt or single step the microprocessor on all cycles. A negative transition to the low state during or coincident with phase one (ϕ_1) will halt the microprocessor with the output address lines reflecting the current address being accessed. During a Write cycle the data bus will reflect the current data being written.

While RDY is low the CPU is in a low power mode.

Bus Enable (BE)

The BE input allows an external device to tri-state the address, data, and R/W lines by taking this line to a logical zero state.

Interrupt Request (IRQ)

This TTL level input requests that an interrupt sequence begin within the microprocessor. The microprocessor will complete the instruction being executed before recognizing the request. At that time, the interrupt mask bit in the Status Code Register will be examined. If the interrupt mask flag is not set, the microprocessor will begin an interrupt sequence. The Program Counter and Processor Status Register are stored in the stack. The microprocessor will then set the interrupt mask flag high so that no further interrupts may occur. At the end of this cycle, the program counter low will be loaded from address FFFE and program counter high from location FFFF, thus transferring program control to the memory vector located at these addresses. The RDY signal must be in the high state for any interrupt to be recognized. An external pull-up resistor should be used for proper wire-OR operation.

Non-Maskable Interrupt (NMI)

A negative going edge on this input requests that a non-maskable interrupt sequence be generated within the microprocessor.

$\overline{\text{NMI}}$ is an unconditional interrupt. Following completion of the current instruction, the sequence of operations defined for $\overline{\text{IRQ}}$ will be performed, regardless of the state of the interrupt mask flag. The vector address loaded into the program counter, low and high, are locations FFFA and FFFB respectively, thereby transferring program control to the memory vector located at these addresses. The instructions loaded at these locations cause the microprocessor to branch to a non-maskable interrupt routine in memory.

$\overline{\text{NMI}}$ requires an external resistor to V_{CC} for proper wire-OR operations.

Inputs $\overline{\text{IRQ}}$ and $\overline{\text{NMI}}$ are hardware interrupts lines sampled during ϕ_2 (phase 2). They begin the appropriate interrupt routine on the ϕ_1 (phase 1) following the completion of the current instruction.

Set Overflow Flag (S.O.)

A negative going edge on this input sets the overflow bit in the Status Code Register. This signal is sampled on the trailing edge of ϕ_1 and must be externally synchronized.

SYNC

This output line identifies those cycles in which the microprocessor is doing an OP CODE fetch. The SYNC line goes high during ϕ_1 of an OP CODE fetch and stays high for the remainder of that cycle. If the RDY line is pulled low during the ϕ_1 clock pulse in which SYNC went high, the processor will stop in its current state and will remain there until the RDY line goes high. In this manner the SYNC signal can be used to control RDY to cause single instruction execution.

Reset

This input resets or starts the microprocessor from a power down condition. During the time that this line is held low, writing to or from the microprocessor is inhibited. When a positive edge is detected on the input, the microprocessor will immediately begin the reset sequence.

After a system initialization time of six clock cycles, the mask interrupt flag will be set and the microprocessor will load the program counter from the memory vector locations FFFC and FFFD. This is the start location for program control.

This line is a Schmitt trigger input which facilitates the use of an RC network as a power on reset circuit.

Memory Lock ($\overline{\text{ML}}$)

This output may be used by external bus arbitration circuitry to avoid the interruption of read-modify-write instructions. These instructions are ASL, DEC, INC, LSR, RMB, ROR, SMB, TRB, and TSB.

ADDRESSING MODES

ACCUMULATOR ADDRESSING—This form of addressing is represented with a one byte instruction, implying an operation on the accumulator.

IMMEDIATE ADDRESSING—In immediate addressing, the second byte of the instruction contains the operand, with no further memory addressing required.

ABSOLUTE ADDRESSING—In absolute addressing, the second byte of the instruction specifies the eight low order bits of the effective address while the third byte specifies the eight high order bits. Thus, the absolute addressing mode allows access to the entire 64K bytes of addressable memory.

ZERO PAGE ADDRESSING—The zero page instructions allow for shorter code and execution times by fetching only the second byte of the instruction and assuming a zero high address byte. Careful use of the zero page can result in significant increase in code efficiency.

INDEXED ZERO PAGE ADDRESSING—(X, Y indexing)—This form of addressing is used with the index register and is referred to as "Zero Page, X" or "Zero Page, Y". The effective address is calculated by adding the second byte to the contents of the index register. Since this is a form of "Zero Page" addressing, the content of the second byte references a location in page zero. Additionally, due to the "Zero Page" addressing nature of this mode, no carry is added to the high order eight bits of memory and crossing of page boundaries does not occur.

INDEXED ABSOLUTE ADDRESSING—(X, Y indexing)—This form of addressing is used in conjunction with X and Y index register and is referred to as "Absolute, X" and "Absolute, Y". The effective address is formed by adding the contents of X or Y to the address contained in the second and third bytes of the instruction. This mode allows the index register to contain the index or count value and the instruction to contain the base address. This type of indexing allows any location referencing and the index to modify multiple fields, resulting in reduced coding and execution time.

INDEXED ABSOLUTE INDIRECT—(new addressing mode—JMP (IND), X)—The contents of the second and third instruction bytes are added to the X-register. The sixteen-bit result is a memory address containing the effective address.

IMPLIED ADDRESSING—In the implied addressing mode, the address containing the operand is implicitly stated in the operation code of the instruction.

RELATIVE ADDRESSING—Relative addressing is used only with branch instructions and establishes a destination for the conditional branch.

The second byte of the instruction becomes the operand which is an "Offset" added to the contents of the lower eight bits of the program counter when the counter is set at the next instruction. The range of the offset is -128 to +127 bytes from the next instruction.

INDEXED INDIRECT ADDRESSING—In indexed indirect addressing (referred to as (Indirect, X)), the second byte of the instruction is added to the contents of the X index register, discarding the carry. The result of this addition points to a memory location on page zero whose contents are the low order eight bits of the effective address. The next memory location in page zero contains the high order eight bits of the effective address. Both memory locations specifying the high and low order bytes of the effective address must be in page zero.

INDIRECT INDEXED ADDRESSING—In indirect indexed addressing (referred to as (Indirect), Y), the second byte of the instruction points to a memory location in page zero. The contents of this memory location are added to the contents of the Y index register, the result being the low order eight bits of the effective address. The carry from this addition is added to the contents of the next page zero memory location, the result being the high order eight bits of the effective address.

ABSOLUTE INDIRECT—The second byte of the instruction contains the low order eight bits of a memory location. The high order eight bits of that memory location are contained in the third byte of the instruction. The contents of the fully specified memory location are the low order byte of the effective address. The next memory location contains the high order byte of the effective address which is loaded into the sixteen bits of the program counter. (JMP (IND) only)

INDIRECT—(new addressing mode)—The second byte of the instruction contains a zero page address serving as the indirect pointer.

INSTRUCTION SET ALPHABETIC SEQUENCE

Mnemonic	Function	Mnemonic	Function
(2) ADC	Add Memory to Accumulator with Carry	NOP	No Operation
(2) AND	"AND" Memory with Accumulator	(2) ORA	"OR" Memory with Accumulator
ASL	Shift Left One Bit (Memory or Accumulator)	PHA	Push Accumulator on Stack
(1) BBR	Branch on Bit Reset	PHP	Push Processor Status on Stack
(1) BBS	Branch on Bit Set	(1) PHX	Push X Register on Stack
BCC	Branch on Carry Clear	(1) PHY	Push Y Register on Stack
BCS	Branch on Carry Set	PLA	Pull Accumulator from Stack
BEQ	Branch on Result Zero	PLP	Pull Processor Status from Stack
(2) BIT	Test Bits in Memory with Accumulator	(1) PLX	Pull X Register from Stack
BMI	Branch on Result Minus	(1) PLY	Pull Y Register from Stack
BNE	Branch on Result not Zero	(1) RMB	Reset Memory Bit
BPL	Branch on Result Plus	ROL	Rotate One Bit Left (Memory or Accumulator)
(1) BRA	Branch Always	ROR	Rotate One Bit Right (Memory or Accumulator)
BRK	Force Break	RTI	Return from Interrupt
BVC	Branch on Overflow Clear	RTS	Return from Subroutine
BVS	Branch on Overflow Set	SBC	Subtract Memory from Accumulator with Borrow
CLC	Clear Carry Flag	SEC	Set Carry Flag
CLD	Clear Decimal Mode	SED	Set Decimal Mode
CLI	Clear Interrupt Disable Bit	SEI	Set Interrupt Disable Status
CLV	Clear Overflow Flag	(1) SMB	Set Memory Bit
(2) CMP	Compare Memory and Accumulator	(2) STA	Store Accumulator in Memory
CPX	Compare Memory and Index X	STX	Store Index X in Memory
CPY	Compare Memory and Index Y	STY	Store Index Y in Memory
(2) DEC	Decrement Memory by One	(1) STZ	Store Zero
DEX	Decrement Index X by One	TAX	Transfer Accumulator to Index X
DEY	Decrement Index Y by One	TAY	Transfer Accumulator to Index Y
(2) EOR	"Exclusive-OR" Memory with Accumulator	(1) TRB	Test and Reset Bits
(2) INC	Increment Memory by One	(1) TSB	Test and Set Bits
INX	Increment Index X by One	TSX	Transfer Stack Pointer to Index X
INY	Increment Index Y by One	TXA	Transfer Index X to Accumulator
(2) JMP	Jump to New Location	TXS	Transfer Index X to Stack Register
JSR	Jump to New Location Saving Return Address	TYA	Transfer Index Y to Accumulator
(2) LDA	Load Accumulator with Memory		
LDX	Load Index X with Memory		
LDY	Load Index Y with Memory		
LSR	Shift One Bit Right (Memory or Accumulator)		

NOTES:

(1) New Instruction

(2) Previous Instruction with additional addressing mode(s)

R65C02, R65C102, R65C112 Microprocessors

INSTRUCTION SET OP CODE MATRIX

MSD	LSD	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
0		BRK Implied 1 7	ORA (IND, X) 2 6			TSB ZP 2 5	ORA ZP 2 3	ASL ZP 2 5	RMB0 ZP 2 5	PHP Implied 1 3	ORA IMM 2 2	ASL Accum 1 2		TSB ABS 3 6	ORA ABS 3 4	ASL ABS 3 6	BBR0 ZP 3 5**	0
1		BPL Relative 2 2**	ORA (IND), Y 2 5*	ORA (IND) 2 5		TRB ZP 2 5	ORA ZP, X 2 4	ASL ZP, X 2 6	RMB1 ZP 2 5	CLC Implied 1 2	ORA ABS, Y 3 4*	INC Accum 1 2		TRB ABS 3 6	ORA ABS, X 3 4*	ASL ABS, X 3 7	BBR1 ZP 3 5**	1
2		JSR Absolute 3 6	AND (IND, X) 2 6			BIT ZP 2 3	AND ZP 2 3	ROL ZP 2 5	RMB2 ZP 2 5	PLP Implied 1 4	AND IMM 2 2	ROL Accum 1 2		BIT ABS 3 4	AND ABS 3 4	ROL ABS 3 6	BBR2 ZP 3 5**	2
3		BMI Relative 2 2**	AND (IND), Y 2 5*	AND (IND) 2 5		BIT ZP, X 2 4	AND ZP, X 2 4	ROL ZP, X 2 6	RMB3 ZP 2 5	SEC Implied 1 2	AND ABS, Y 3 4*	DEC Accum 1 2		BIT ABS, X 3 4*	AND ABS, X 3 4*	ROL ABS, X 3 7	BBR3 ZP 3 5**	3
4		RTI Implied 1 6	EOR (IND, X) 2 6				EOR ZP 2 3	LSR ZP 2 5	RMB4 ZP 2 5	PHA Implied 1 3	EOR IMM 2 2	LSR Accum 1 2		JMP ABS 3 3	EOR ABS 3 4	LSR ABS 3 6	BBR4 ZP 3 5**	4
5		BVC Relative 2 2**	EOR (IND), Y 2 5*	EOR (IND) 2 5			EOR ZP, X 2 4	LSR ZP, X 2 6	RMB5 ZP 2 5	CLI Implied 1 2	EOR ABS, Y 3 4*	PHY Implied 1 2			EOR ABS, X 3 4*	LSR ABS, X 3 7	BBR5 ZP 3 5**	5
6		RTS Implied 1 6	ADC (IND, X) 2 6†			STZ ZP 2 3	ADC ZP 2 3†	ROR ZP 2 5	RMB6 ZP 2 5	PLA Implied 1 4	ADC IMM 2 2†	ROR Accum 1 2		JMP Indirect 3 5	ADC ABS 3 4†	ROR ABS 3 6	BBR6 ZP 3 5**	6
7		BVS Relative 2 2**	ADC (IND), Y 2 5†	ADC (IND) 2 5†		STZ ZP, X 2 4	ADC ZP, X 2 4†	ROR ZP, X 2 6	RMB7 ZP 2 5	SEI Implied 1 2	ADC ABS, Y 3 4†	PLY Implied 1 2		JMP (IND), X 3 6	ADC ABS, X 3 4†	ROR ABS, X 3 7	BBR7 ZP 3 5**	7
8		BRA Relative 2 3	STA (IND, X) 2 6			STY ZP 2 3	STA ZP 2 3	STX ZP 2 3	SMB0 ZP 2 5	DEY Implied 1 2	BIT IMM 2 2	TXA Implied 1 2		STY ABS 3 4	STA ABS 3 4	STX ABS 3 4	BBR8 ZP 3 5**	8
9		BCC Relative 2 2**	STA (IND), Y 2 6	STA (IND) 2 6		STY ZP, X 2 4	STA ZP, X 2 4	STX ZP, Y 2 4	SMB1 ZP 2 5	TYA Implied 1 2	STA ABS, Y 3 5	TXS Implied 1 2		STZ ABS 3 4	STA ABS, X 3 5	STZ ABS, X 3 5	BBR9 ZP 3 5**	9
A		LDY IMM 2 2	LDA (IND, X) 2 6	LDX IMM 2 2		LDY ZP 2 3	LDA ZP 2 3	LDX ZP 2 3	SMB2 ZP 2 5	TAY Implied 1 2	LDA IMM 2 2	TAX Implied 1 2		LDY ABS 3 4	LDA ABS 3 4	LDX ABS 3 4	BBR10 ZP 3 5**	A
B		BCS Relative 2 2**	LDA (IND), Y 2 5*	LDA (IND) 2 5		LDY ZP, X 2 4	LDA ZP, X 2 4	LDX ZP, Y 2 4	SMB3 ZP 2 5	CLV Implied 1 2	LDA ABS, Y 3 4*	TSX Implied 1 2		LDY ABS, X 3 4*	LDA ABS, X 3 4*	LDX ABS, Y 3 4*	BBR11 ZP 3 5**	B
C		CPY IMM 2 2	CMP (IND, X) 2 6			CPY ZP 2 3	CMP ZP 2 3	DEC ZP 2 5	SMB4 ZP 2 5	INY Implied 1 2	CMP IMM 2 2	DEX Implied 1 2		CPY ABS 3 4	CMP ABS 3 4	DEC ABS 3 6	BBR12 ZP 3 5**	C
D		BNE Relative 2 2**	CMP (IND), Y 2 5*	CMP (IND) 2 5			CMP ZP, X 2 4	DEC ZP, X 2 6	SMB5 ZP 2 5	CLD Implied 1 2	CMP ABS, Y 3 4*	PHX Implied 1 2			CMP ABS, X 3 4*	DEC ABS, X 3 7	BBR13 ZP 3 5**	D
E		CPX IMM 2 2	SBC (IND, X) 2 6†			CPX ZP 2 3	SBC ZP 2 3†	INC ZP 2 5	SMB6 ZP 2 5	INX Implied 1 2	SBC IMM 2 2†	NOP Implied 1 2		CPX ABS 3 4	SBC ABS 3 4†	INC ABS 3 6	BBR14 ZP 3 5**	E
F		BEQ Relative 2 2**	SBC (IND), Y 2 5†	SBC (IND) 2 5†			SBC ZP, X 2 4†	INC ZP, X 2 6	SMB7 ZP 2 5	SED Implied 1 2	SBC ABS, Y 3 4†	PLX Implied 1 2			SBC ABS, X 3 4†	INC ABS, X 3 7	BBR15 ZP 3 5**	F
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	



— New Opcode

0	BRK Implied 1 7
---	-----------------------

—OP Code
—Addressing Mode
—Instruction Bytes; Machine Cycles

†Add 1 to N if in decimal mode.

*Add 1 to N if page boundary is crossed.

**Add 1 to N if branch occurs to same page;
Add 2 to N if branch occurs to different page.

INSTRUCTION SUMMARY

6

R65C02, R65C102, R65C112 Microprocessors

HARDWARE SPECIFICATIONS

R65C02—40 Pin Package

Pin Outs

VSS	1	40	RES
RDY	2	39	ϕ_2 (OUT)
ϕ_1 (OUT)	3	38	S.O.
IRQ	4	37	ϕ_0 (IN)
N.C.	5	36	N.C.
NMI	6	35	N.C.
SYNC	7	34	R/W
VCC	8	33	D0
A0	9	32	D1
A1	10	31	D2
A2	11	30	D3
A3	12	29	D4
A4	13	28	D5
A5	14	27	D6
A6	15	26	D7
A7	16	25	A15
A8	17	24	A14
A9	18	23	A13
A10	19	22	A12
A11	20	21	VSS

FEATURES

- Pin Compatible with NMOS R6502
- 64K Addressable Bytes of Memory (A0-A15)
- IRQ Interrupt
- On-the-chip Clock
 - TTL Level Single Phase Input
- SYNC Signal
 - (can be used for single instruction execution)
- RDY Signal
 - (can be used to halt or single cycle execution)
- Two Phase Output Clock for Timing of Support Chips
- NMI Interrupt

R65C102—40 Pin Package

VSS	1	40	RES
RDY	2	39	ϕ_2 (OUT)
ϕ_1 (OUT)	3	38	S.O.
IRQ	4	37	XTLI
ML	5	36	BE
NMI	6	35	XTLO
SYNC	7	34	R/W
VCC	8	33	D0
A0	9	32	D1
A1	10	31	D2
A2	11	30	D3
A3	12	29	D4
A4	13	28	D5
A5	14	27	D6
A6	15	26	D7
A7	16	25	A15
A8	17	24	A14
A9	18	23	A13
A10	19	22	A12
A11	20	21	VSS

FEATURES

- ϕ_1 Quadrature Clock Output eases access time requirements
- 64K Addressable Bytes of Memory (A0-A15)
- IRQ Interrupt
- On-the-chip Clock
 - TTL Level Single Phase Input
 - RC Time Base Input
 - Crystal Time Base Input ($\div 4$)
- SYNC Signal (can be used for signal instruction execution)
- RDY Signal (can be used to halt or single cycle execution)
- Two Phase Output Clock for Timing of Support Chips
- NMI Interrupt
- Direct Memory Access Capability
- Memory Lock Output
- Bus Enable Signal

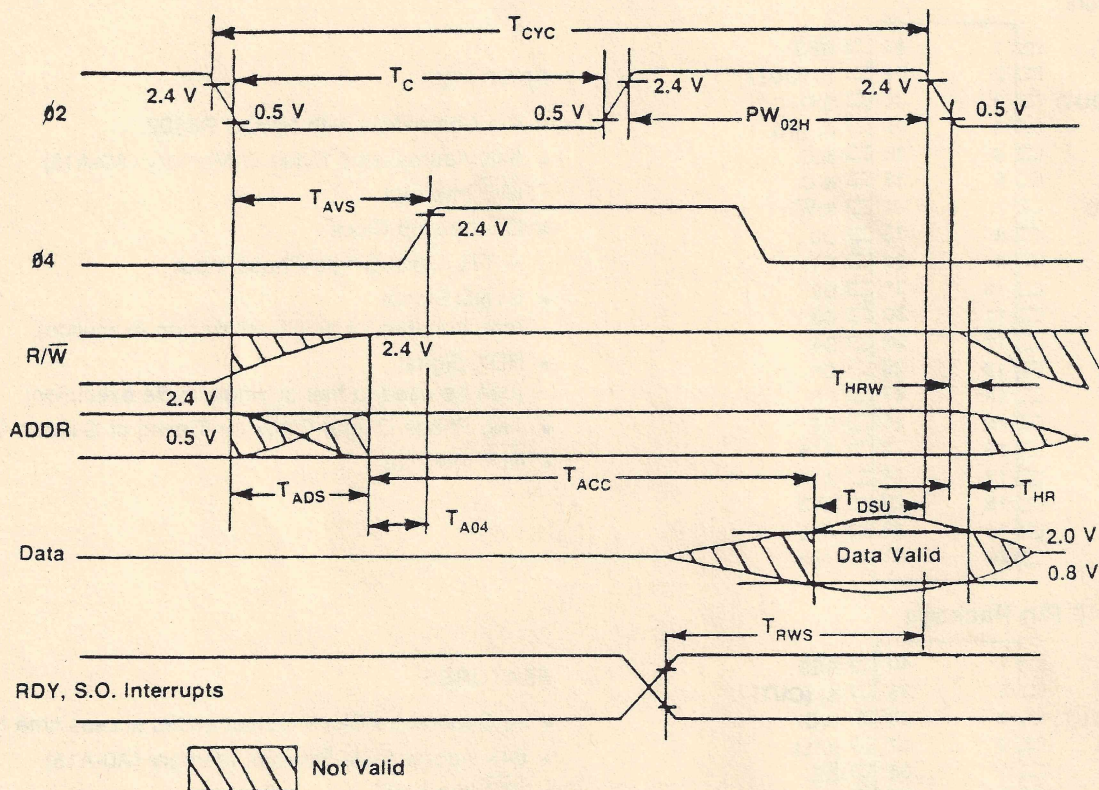
R65C112—40 Pin Package

VSS	1	40	RES
RDY	2	39	N.C.
N.C.	3	38	S.O.
IRQ	4	37	ϕ_2 (IN)
ML	5	36	BE
NMI	6	35	N.C.
SYNC	7	34	R/W
VCC	8	33	D0
A0	9	32	D1
A1	10	31	D2
A2	11	30	D3
A3	12	29	D4
A4	13	28	D5
A5	14	27	D6
A6	15	26	D7
A7	16	25	A15
A8	17	24	A14
A9	18	23	A13
A10	19	22	A12
A11	20	21	VSS

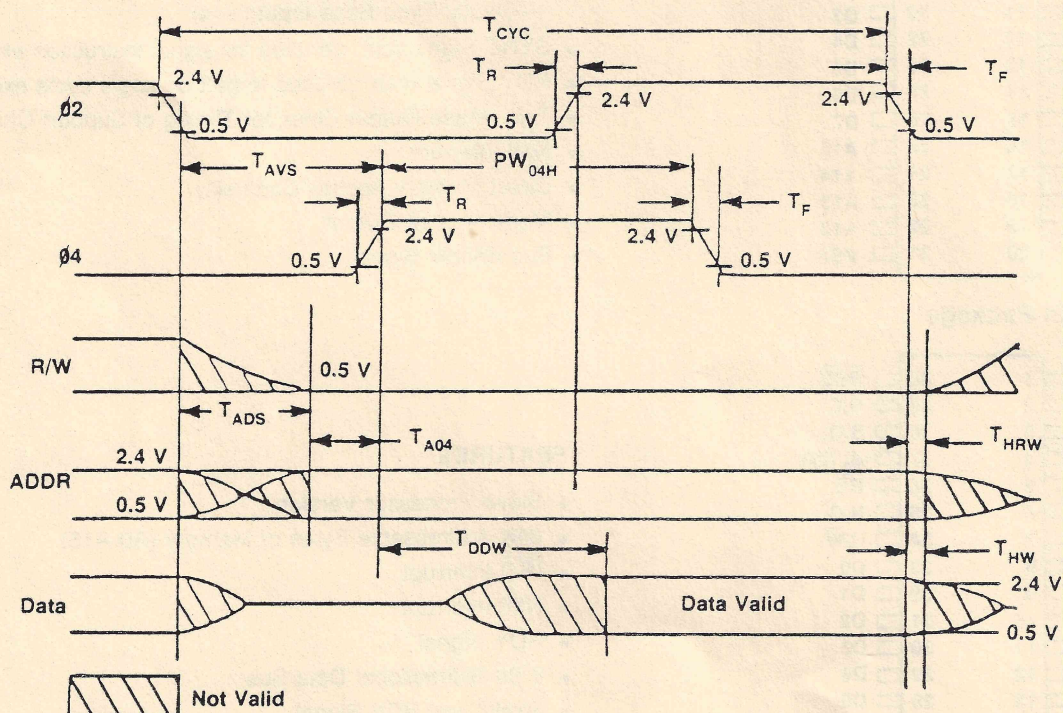
FEATURES

- Slave Processor Version
- 64K Addressable Bytes of Memory (A0-A15)
- IRQ Interrupt
- NMI Interrupt
- RDY Signal
- 8 Bit Bidirectional Data Bus
- SYNC and RDY Signal
- Two phase clock input
- Bus Enable
- Direct Memory Access capability
- Memory Lock Output

READ DATA FROM MEMORY OR PERIPHERALS TIMING



WRITE DATA TO MEMORY OR PERIPHERALS TIMING



*Hold time for BA, BS not specified

R65C02, R65C102, R65C112 Microprocessors

A.C. Electrical Timing Characteristics

Characteristic	Symbol	2MHz		3MHz		4MHz		Units
		Min	Max	Min	Max	Min	Max	
Cycle Time	T_{CYC}	500		333		250		ns
Pulse Width, 02 Low	PW_{02L}	210		160		100		ns
Pulse Width, 02 High	PW_{02H}	220		170		110		ns
Clock Rise & Fall Time	T_R, T_F		15		12		10	ns
Pulse Width, 04 Low	PW_{04L}	210		150		100		ns
Pulse Width, 04 High	PW_{04H}	220		160		110		ns
Delay Time 02 to 04 Rise	T_{AVS}	80	125		94		63	ns
Address Delay	T_{ADS}		100		75		50	ns
Address Hold Time (Address, R/W)	T_{HRW}	20		20		20		ns
Address Valid to 04 Rise	T_{A04}	25		18		12		ns
Data Delay Time (Write)	T_{ODW}		110		82		55	ns
Read Data Setup Time	T_{DSU}	40		30		20		ns
Read Data Hold Time	T_{HR}	10		10		10		ns
Write Data Hold Time	T_{HW}	30		30		30		ns
Read Access Time	T_{ACC}	340		254		168		ns
Processor Control Setup Time (RDY, S.O. Interrupts, Reset)	T_{RWS}	110		80		60		ns
Bus Enable Setup Time	T_{BE}	125		100		75		ns

R65C02, R65C102, R65C112 Microprocessors

D.C. CHARACTERISTICS

Maximum Ratings

Rating	Symbol	Value	Unit
Supply Voltage	V_{CC}	-0.3 to +7.0	Vdc
Input Voltage	V_{in}	-0.3 to +7.0	Vdc
Operating Temperature	T		°C
Commercial		0 to +70	
Industrial		-40 to +85	
Storage Temperature	T_{STG}	-55 to +150	°C

NOTE

This device contains input protection against damage to high static voltages or electric fields; however, precautions should be taken to avoid application of voltages higher than maximum rating.

Electrical Characteristics

($V_{CC} = 5.0 \pm 20\%$, $V_{SS} = 0$)

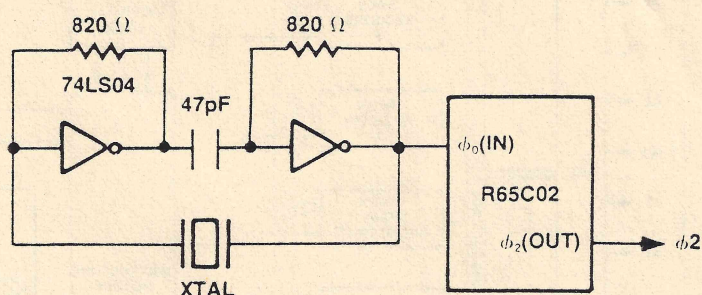
Characteristic	Symbol	Min	Max	Unit
Input High Voltage All Input Pins (except ϕ_2 on R65C112)	V_{IH}	2.0	$V_{CC} + 0.3$	Vdc
Input Low Voltage All Input Pins (except ϕ_2 on R65C112)	V_{IL}	-0.3	0.8	Vdc
Input High Voltage ϕ_2 in on R65C112	V_{IH}	2.4	—	Vdc
Input Low Voltage ϕ_2 in on R65C112	V_{IL}	—	0.4	Vdc
Input Leakage Current ($V_{in} = 0$ to 5.25V, $V_{CC} = 0$) Logic (Excl. Rdy, S.O.) ϕ_1, ϕ_2 $\phi_{o(in)}$	I_{in}	— — —	1.0 1.0 1.0	μA
Three-State (Off State) Input Current ($V_{in} = 0.4$ to 2.4V, $V_{CC} = 5.25V$) Data Lines	I_{TSI}	—	10	μA
Output High Voltage ($I_{LOAD} = -100 \mu A$, $V_{CC} = 4.75V$) SYNC, Data, A0-A15, $\overline{R/W}$, ϕ_1, ϕ_2	V_{OH}	$V_{SS} + 2.4$	—	Vdc
Output Low Voltage ($I_{LOAD} = 1.6 \text{ mA}$, $V_{CC} = 4.75V$) SYNC, Data, A0-A15, $\overline{R/W}$, ϕ_1, ϕ_2	V_{OL}		$V_{SS} + 0.4$	Vdc
Power Dissipation 0 MHz (Standby) 1 MHz 2 MHz 3 MHz 4 MHz Low Power (RDY = 0)	P_D	— — — — —	10 20 40 60 80 10	μW mW mW/MHz
Capacitance at 25°C ($V_{in} = 0$, $f = 1 \text{ MHz}$) Logic Data A0-A15, $\overline{R/W}$, SYNC $\phi_{o(in)}$ ϕ_1 ϕ_2	C C_{in} C_{out} $C\phi_{o(in)}$ $C\phi_1$ $C\phi_2$	— — — — — —	5 10 10 10 30 50	pF

NOTE

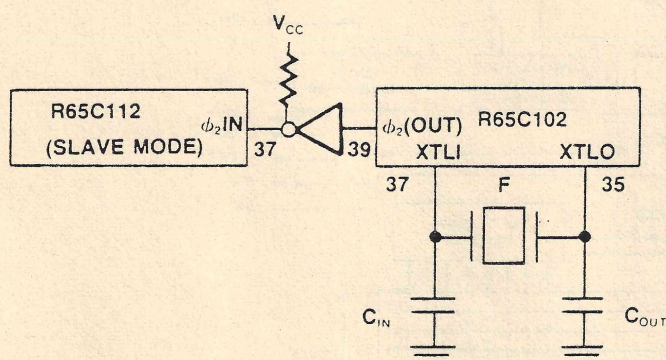
$\overline{IRQ}$ and $\overline{NMI}$ require external pull-up resistor.

CLOCK CONSIDERATIONS

EXAMPLE TIME BASE GENERATION



*CRYSTAL: CTS KNIGHTS MP SERIES, OR EQUIVALENT



F	CIN	COUT	ϕ_2
16 MHZ	16PF	16PF	4 MHZ
8 MHZ	18PF	18PF	2 MHZ
6 MHZ	20PF	20PF	1.5 MHZ
4 MHZ	24PF	24PF	1 MHZ

The oscillator in the R65C102 is series resonant.

The crystal input is divided by 4: (R65C102 ONLY)

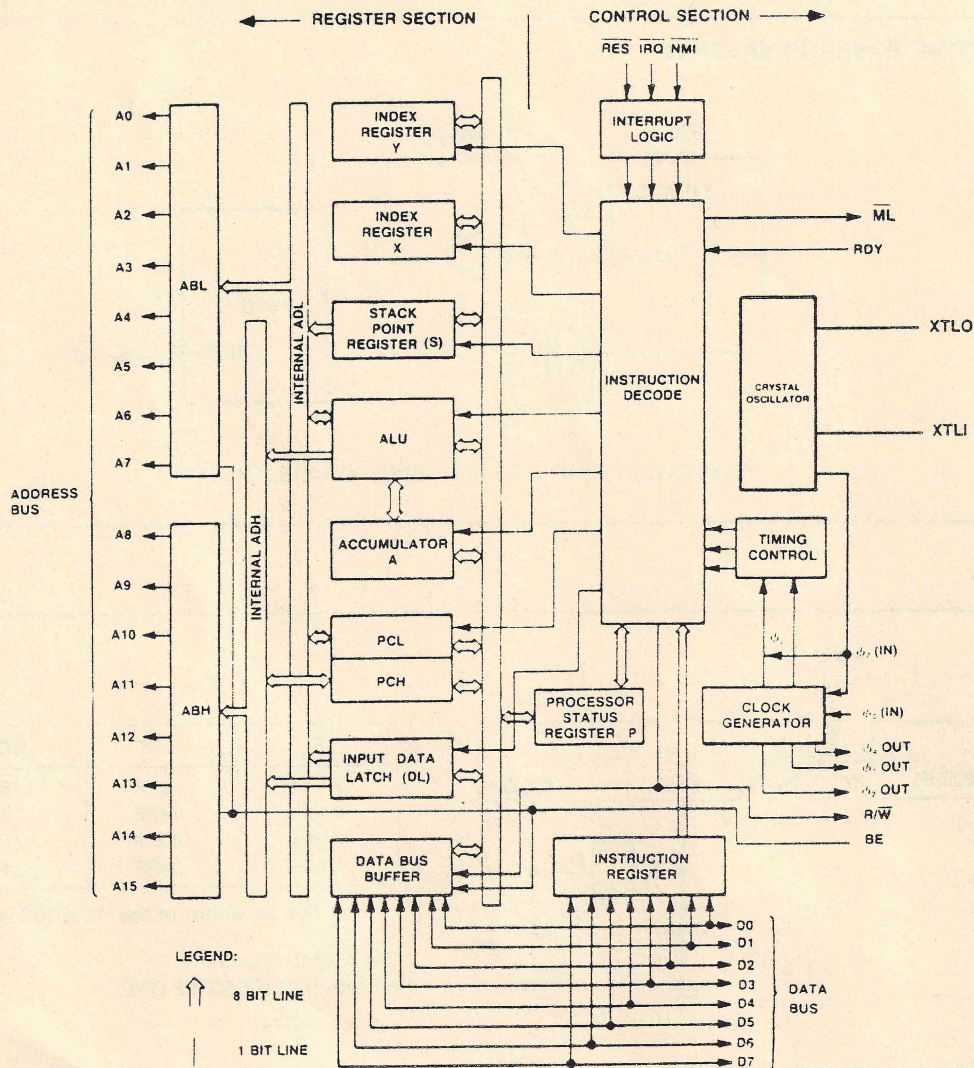
$$\phi_2 = \frac{XTAL}{4}$$

NOMINAL CRYSTAL PARAMETERS

	3.58	4.0	6.0	8.0	16.0	MHZ
RS	60	50	30-50	20-40	10-30	Ω
C0	3.5	6.5	4-6	4-6	3-5	PF
C1	.015	.025	.01-.02	.01-.02	.01-.02	PF
Q	740K	730K	720K	720K	720K	K

Note: These represent at-cut crystal parameters only. Others may be used.

R6500 INTERNAL ARCHITECTURE



ELECTRONIC DEVICES DIVISION REGIONAL ROCKWELL SALES OFFICES

HOME OFFICE

Electronic Devices Division
Rockwell International
4311 Jamboree Road
Newport Beach, California 92660

Mailing Address:

P.O. Box C
Newport Beach, California 92660
Mail Code 501-300
Tel: 714-833-4700
TWX: 910 591-1698

UNITED STATES

Electronic Devices Division
Rockwell International
1842 Reynolds
Irvine, California 92714
(714) 833-4655
ESL 62108710
TWX: 910 595-2518

Electronic Devices Division
Rockwell International
921 Bowser Road
Richardson, Texas 75080
(214) 996-6500
Telex: 73-307

Electronic Devices Division
Rockwell International
10700 West Higgins Rd., Suite 102
Rosemont, Illinois 60018
(312) 297-8862
TWX: 910 233-0179 (RI MED ROSM)

Electronic Devices Division
Rockwell International
5001B Greentree
Executive Campus, Rt. 73
Marlton, New Jersey 08053
(609) 596-0090
TWX: 710 940-1377

FAR EAST

Electronic Devices Division
Rockwell International Overseas Corp.
Ithopia Hirakawa-cho Bldg.
7-6, 2-chome, Hirakawa-cho
Chiyoda-ku, Tokyo 102, Japan
(03) 265-8806
Telex: J22198

EUROPE

Electronic Devices Division
Rockwell International GmbH
Fraunhoferstrasse 11
D-8033 Munchen-Martinsried
West Germany
(089) 857-6016
Telex: 0521/2650 rimd d

Electronic Devices Division
Rockwell International
Heathrow House, Bath Rd.
Cranford, Hounslow,
Middlesex, England
(01) 759-9911
Telex: 851-25463

Electronic Devices
Rockwell Collins
Via Boccaccio, 23
20123 Milano, Italy
498.74.79
Telex: 202182

YOUR LOCAL REPRESENTATIVE

